НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Веб-додаток для забезпечення безпеки руху пішоходів»

Виконала:
студентка IV курсу, групи ІО-62
Медведкова Юлія Іллівна _____

Керівник:
к.т.н, доцент
Русанова О.В. _____
Консультант з нормоконтролю:
Професор, д.т.н.,
Сімоненко В.П. _____

Рецензент:
Доц. каф. СПКС, к. т. н.
Марія Миколаївна Орлова _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут»

Інститут (факультет) інформатики та обчислювальної техніки
(повна назва)
Кафедра обчислювальної техніки
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 6.050102 – « Комп'ютерна інженерія»
(повна назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ *Сергій Стіренко*
"___" _____ 2020 року

ЗАВДАННЯ
на дипломний проект студента

Медведкова Юлія Іллівна
(прізвище, ім'я, по батькові)

1. Тема проекту *«Веб-додаток для забезпечення безпеки руху пішоходів»*

керівник проекту *к.т.н, доцент Русанова О.В.*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "_07_" _травня____ 2020_ року №1081-с

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту технічна документація, теоретичні дані, інтернет-публікації за темою роботи

4. Зміст пояснювальної записки

−Аналіз предметної області;
−Огляд існуючих додатків з схожою концепцією;
− Огляд засобів розробки;
− Проектування веб-додатку;
− Реалізація веб-додатку та тестування.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)
___ Алгоритмічна схема, функціональна схема та структурна схема _____
6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	*д.т.н., проф. Сімоненко В.П.*		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	*Затвердження теми роботи*	*1.09.2019*	
2	*Вивчення та аналіз завдання*	*2.09.2019-12.03.2020*	
3	*Визначення вхідних та вихідних даних, формулювання алгоритму*	*12.03.2020-22.03.2020*	
4	*Проведення аналізу засобів для розробки програмного забезпечення*	*22.03.2020-2.04.2020*	
5	*Розробка окремих інтерфейсів програми*	*2.04.2020-13.04.2020*	
6	*Проведення моделювання та аналізу розробленого методу*	*13.04.2020-21.05.2020*	
7	*Оформлення матеріалів роботи*	*21.05.2020 – 25.05.2020*	
8	*Передзахист*	*26.05.2020*	
9	*Захист*		

Студент _____ Юлія МЕДВЕДКОВА
(підпис) (ім’я та прізвище)

Керівник проекту (роботи) _____ Ольга РУСАНОВА
(підпис) (ім’я та прізвище)

Анотація

У дипломній роботі на основі фреймворку Spring реалізовано веб-додаток для забезпечення безпеки руху пішоходів. В наш час є дуже актуальна тема безпеки пішоходів, оскільки щодня пішоходи стикаються з незаконними тимчасовими об'єктами, припаркованих транспортних засобів і рушійних транспортних засобів.

Веб-додаток спрощує подачу заявок про порушення, а також з інструментом «пошук» спрощує відображення заявок для користувачів з правами адміністратора.

Аннотация

В дипломной работе на основе фреймворка Spring реализовано веб-приложение для обеспечения безопасности движения пешеходов. В наше время является очень актуальная тема безопасности пешеходов, поскольку ежедневно пешеходы сталкиваются с незаконными временными объектами, припаркованных транспортных средств и движущих транспортных средств.

Веб-приложение упрощает подачу заявок о нарушениях, а также с инструментом «поиск» упрощает отображение заявок для пользователей с правами администратора.

Abstract

Based on the Spring framework, a web application for ensuring pedestrian traffic safety is implemented. Nowadays, the topic of pedestrian safety is a very topical one, as pedestrians are faced with illegal temporary objects, parked vehicles and moving vehicles on a daily basis.

The web application simplifies the filing of complaints about violations, and also with the "search" tool simplifies the display of applications for users with administrator rights.

№ рядка	Формат	Позначення	Найменування	Кількість	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.006515.001 ВП	Відомість дипломного проекту	1	
3	А4	ІАЛЦ.006515.002 ТЗ	Технічне завдання	3	
4	А4	ІАЛЦ.006515.003 ПЗ	Пояснювальна записка	61	
5	А4	ІАЛЦ.006515.004 Д1	Алгоритм схема	1	
6	А4	ІАЛЦ.006515.005 Д2	Функціональна схема	1	
7	А3	ІАЛЦ.006515.006 Д3	Структурна схема	1	

Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.006515.001 ВП		
Розробив		Медведкова Ю.і,			Веб-додаток для безпеки руху пішоходів	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Русанова О.В.					
Н. контр.		Симоненко В.П.			Відомість дипломного проекту	НТУУ “КПІ”, ФІОТ, ІО-62	
Затверд.							

Технічне завдання до дипломного проекту

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку веб-додатку за допомогою фреймворка Spring.

Область застосування: захист руху пішоходів на тротуарі.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки веб-додатку для забезпечення руху пішоходів, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного дипломного проекту є розробка веб-додатка для спрощення подачі заяв за порушення.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп'ютерних технологій, публікації в періодичних виданнях, довідники з розробки мобільних додатків, публікації в Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Розробка інтерфейсу для створення та відображення заявок.
- Виконання розбиття графу алгоритму на яруси.
- Виконання програмної емуляції роботи алгоритму адаптивної реконфігурації
- Розробка засобів візуалізації результатів моделювання.

5.2. Вимоги до програмного забезпечення

- Наявність інтернету та браузер

Пояснювальна записка
до дипломного проекту

на тему: Веб-додаток для безпеки руху пішоходів

Київ - 2020 року

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

MVC (Model View Controller) - Модель-Вид-Контролер;

ПДР - Правила дорожнього руху

Backend – розробка серверної частини;

Frontend – розробка частини інтерфейсу.

ВСТУП

21 століття - століття цифрових технологій. Ні для кого не секрет, що цифрові технології увійшли в наше життя. Більшість людей не може уявити своє існування без них. Вони швидко розвиваються і змінюють наш навколишній світ. Цифрові технології призначені для більш швидкої та простої передачі даних.

Актуальність теми

Кількість нових користувачів Інтернету зростає кожним днем і це не дивно, адже в всесвітній павутині розташовується велика кількість веб-сторінок, які несуть за собою величезну кількість інформації. Користувачі всесвітньої павутини часто виходять з офлайну в онлайн. Можливість зайти на веб-сторінку з комп'ютера, смартфона і планшета дозволяють користувачам щодня відвідувати веб-сторінки. Одні з популярних сторінок, які відвідують українці: пошуковики, відеохостінги, соціальні мережі, електронні пости і блоги.

Оскільки цифрові технології дуже швидко розвиваються, з часом приходить необхідність в створенні нових проектів, які могли б дозволити спростити наше життя і вирішити безліч проблем. Для того, щоб полегшити роботу користувачів в всесвітній павутині, були розроблені веб-додатки. Веб-додатки мають велику функціональність, яка дозволяє поліпшити взаємодію користувачів з продуктом, а також зацікавити і утримати користувача на веб-сторінці. А головна особливість веб-додатків - це те, що користувачеві не потрібно завантажувати додаткових програмних забезпечень. Адже в кожному девайсі є системний браузер і користувачеві необхідно лише підключитися до Інтернету.

Мета і задачі дослідження

Метою роботи є розробка веб-додатку, який би захистив пішоходів на тротуарі від незаконно встановлених об'єктів, незаконно припаркованих

транспортних засобів та руху транспортного засобу. А також, спростити перегляд та перевірку заяв для сторони адміністратору додатку.

Такі основні задачі були поставлені для досягнення мети дипломного проекту:

- Провести аналіз прикладної роботи. Розглянути поняття веб-додатку, визначити призначення дипломного проекту, визначити його переваги та недоліки та зробити постановку задачі;
- Спроектувати дипломний проект. Розробити схему даних та класів, розглянути архітектуру веб-додатків, спроектувати клієнтську та серверну частину додатку, розглянути поняття бази даних та спроектувати базу даних курсового проекту, а також розглянути засоби розробки проекту;
- Реалізувати веб-додаток для безпеки руху пішоходів;
- Провести тестування веб-додатку.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Поняття веб-додатків та веб-сайтів

Спочатку всесвітня павутина складалася виключно з веб-сайтів, які несли за собою тільки інформаційне навантаження. Сайти містили статичні документи і для того, щоб знайти ці документи та відобразити їх на комп'ютері користувача, були створені браузери. Однак, виникла велика проблема - всім користувачам всесвітньої павутини відображалася однакова інформація. Так з часом виникли динамічні сторінки.

Існує два основні види веб-сайту, які визначають поведінку веб-сторінок сайту: статичний і динамічний.

Статичний сайт[1] - це сайт, який складається з незмінних веб-сторінок. Всім користувачам надається однакова інформація, яку користувач не може самостійно змінити. Найчастіше, такі сайти не часто оновляються і на веб-сторінках не доповнюється інформація. Адже для цього розробник повинен самостійно внести зміни до початкового коду веб-сторінки. Цей вид сайту прекрасно підходить для створення сайт-візитки або блогу. Для розробки такого сайту використовується HTML, CSS, JavaScript.

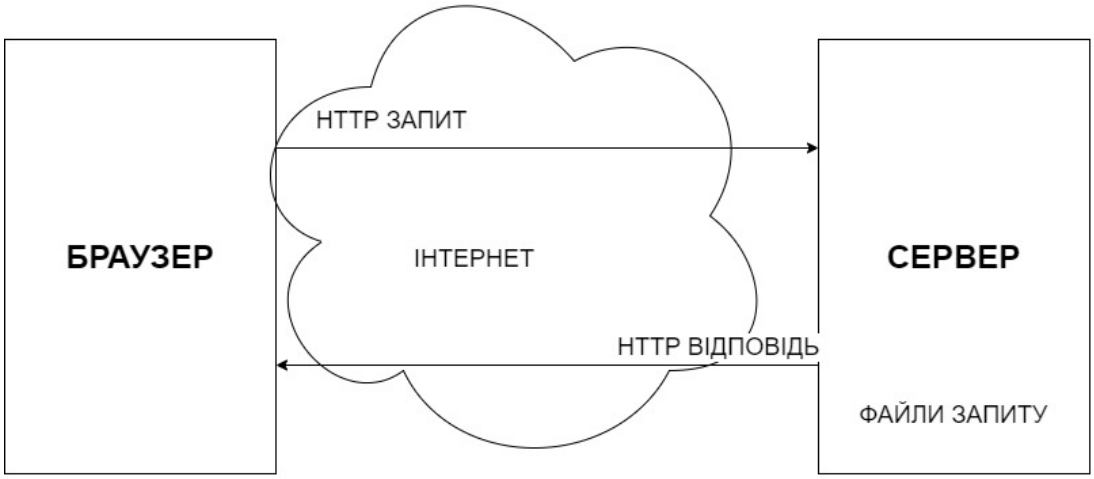


Рис. 1.1. – Схема роботи статичного сайту.

Переваги статичного сайту:

- Швидке завантаження веб-сторінок. Статичні сайти не мають інтерпретатора та вся інформація міститься в одному HTML файлі, який може буди зв’язаний з CSS та JavaScript файлами;
- Вихідні файли сайту займають не так багато місця, тому не повинні виникнути проблеми з оперативною пам'яттю сервера;
- Вихідні файли сайту можна легко перенести на новий сервер;
- Гнучкість. Сторінки можуть відрізнятися один від одної;
- Дешевизна. Вартість розробки статичного веб-сайту значно нижче динамічного веб-сайту.

Недоліки статичного сайту:

- Оновлення контенту. Якщо статичний сайт складається з безлічі сторінок, виникає проблема з доповненням, видаленням і зміною інформації. Якщо прийде необхідність додати новий розділ в меню, тоді розробнику доведеться власноруч ввести поправки в початковий код всіх веб-сторінок;
- Інформація може не оновлюватися і це приводить до того, що користувач не затримується на веб-сторінці;
- Масштабованість. Якщо сайт продажу статичного виду, то з часом може виникнути необхідність в додаванні окремих сторінках з новим товаром. На це буде потрібно більше часу і витрат;
- Витрати на оновлення контенту.

Останнім часом динамічні сайти набрали велику популярність. Веб-сторінки адаптуються під користувача і змінюють свій зовнішній вигляд. Динамічні[2] сторінки мають інтерпретатор, який отримує з веб-сервера документ, генерує код HTML-документа і передає його веб-серверу. Інтерпретатор може буди на будь-якій мові програмування, наприклад: Java, PHP, Ruby. Динамічний сайт використовують в розробці соціальних мереж, інтернет-магазинів, форуми. Роботу такого сайту показано на рис.1.2.

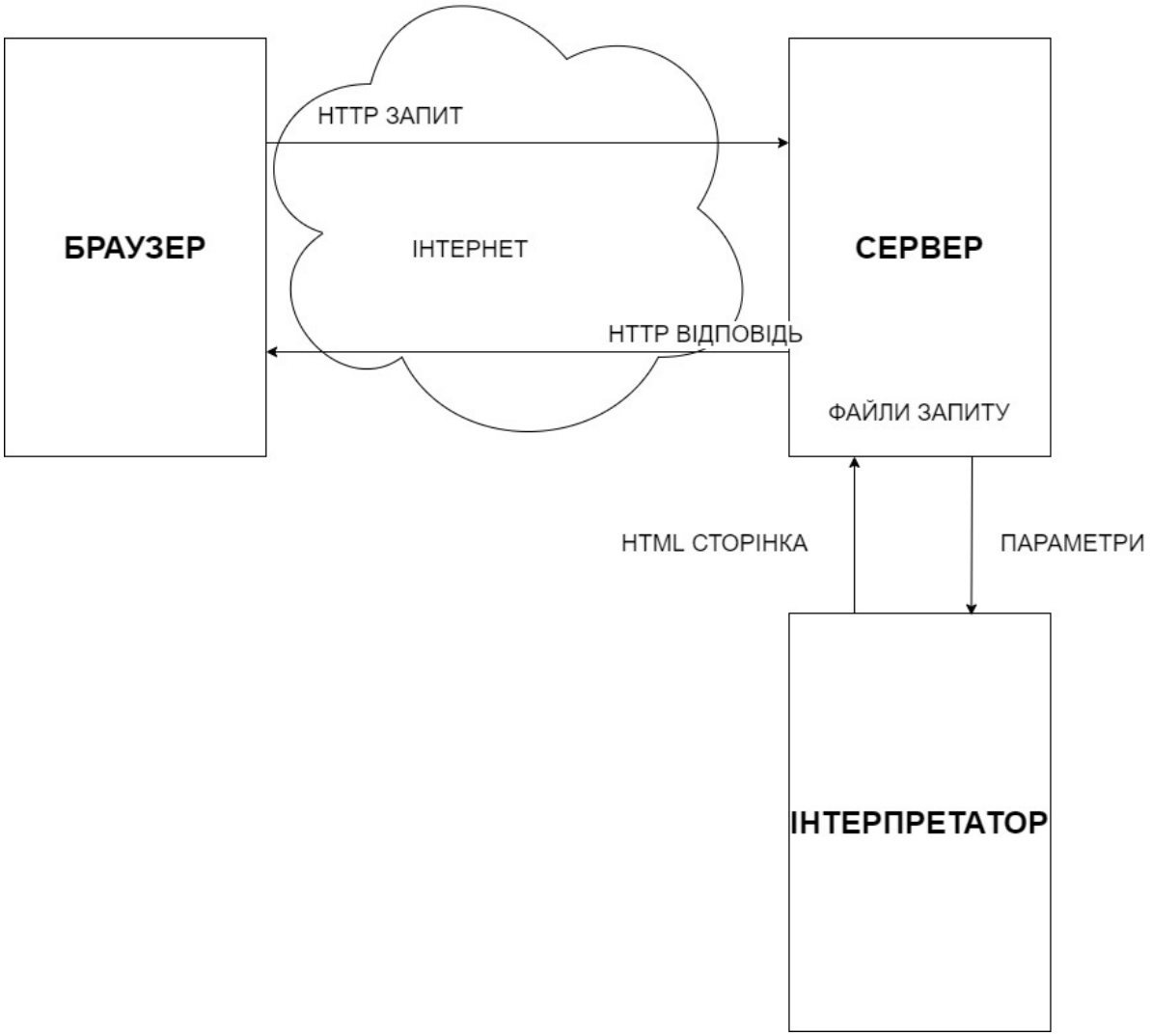


Рис. 1.2. – Схема роботи динамічного сайту.

Переваги динамічного сайту:

- Змінюють вміст веб-сторінки. Веб-сторінки не розміщені на сервері в готовому вигляді. Сервер відправляє документ інтерпретатору, щоб він виконав необхідні йому дії і відправив на сервер згенерований код HTML-документа. Інтерпретатор може взаємодіяти з базою даних, файлової системою або поштовим сервером;
- Функціональність. Дозволяє зробити веб-сайт легким і доступним у використанні;
- Простота в оновлення та підтримки веб-сайту. Розробнику не потрібно змінювати кожну сторінку окремо, як в статичних сайтах;

- Створення складних сторінок;
- Підключення до бази даних. Інформація зберігатися в базі даних, що дає можливість знайти та вилучити інформацію;
- Взаємодія з користувачами.

Недоліки динамічного сайту:

- Повільне завантаження веб-сторінок. Статичний сайт після запиту відразу відправляє HTTP відповідь, але динамічні сайти мають інтерпретатор, який генерує код HTML-документа і відправляє його на сервер. При чому інтерпретатор може ще взаємодіяти з базою даних, поштовим сервером або файлової системою;
- Вихідні файли сайту займають багато місця, тому можуть виникнути проблеми з оперативною пам'яттю сервера;
- Можуть виникнути проблеми з перенесенням на новий сервер;
- Дорожнеча.

Для того, щоб користувачі Інтернету могли бачити різну інформацію, були створені веб-додатки. Вони набрали велику популярність завдяки можливості отримання і обробки інформації від користувача[3]. Інтерактивність дозволяє користувачу брати активну участь в роботі з додатком. Сучасні веб-додатки адаптують сторінку сайту під користувача. Завдяки веб-додаткам, з'явилася можливість реалізовувати безліч корисних функцій. Останнім часом залишаються популярними такі функції: інтернет-магазин, електронна пошта, соціальна мережа, банківська транзакція, пошуковик, аукціон. Веб-додатки доступні для всіх користувачів всесвітньої павутини. На відміну від мобільних додатків, в веб-додаток можна зайти з будь-якого пристрою, у якого є доступ до Інтернету, будь то комп'ютер, смартфон або планшет. Користувачеві не потрібно нічого завантажувати і оновлювати. Це робить веб-додаток дуже доступним.

Роботу такого сайту показано на рис.1.3.

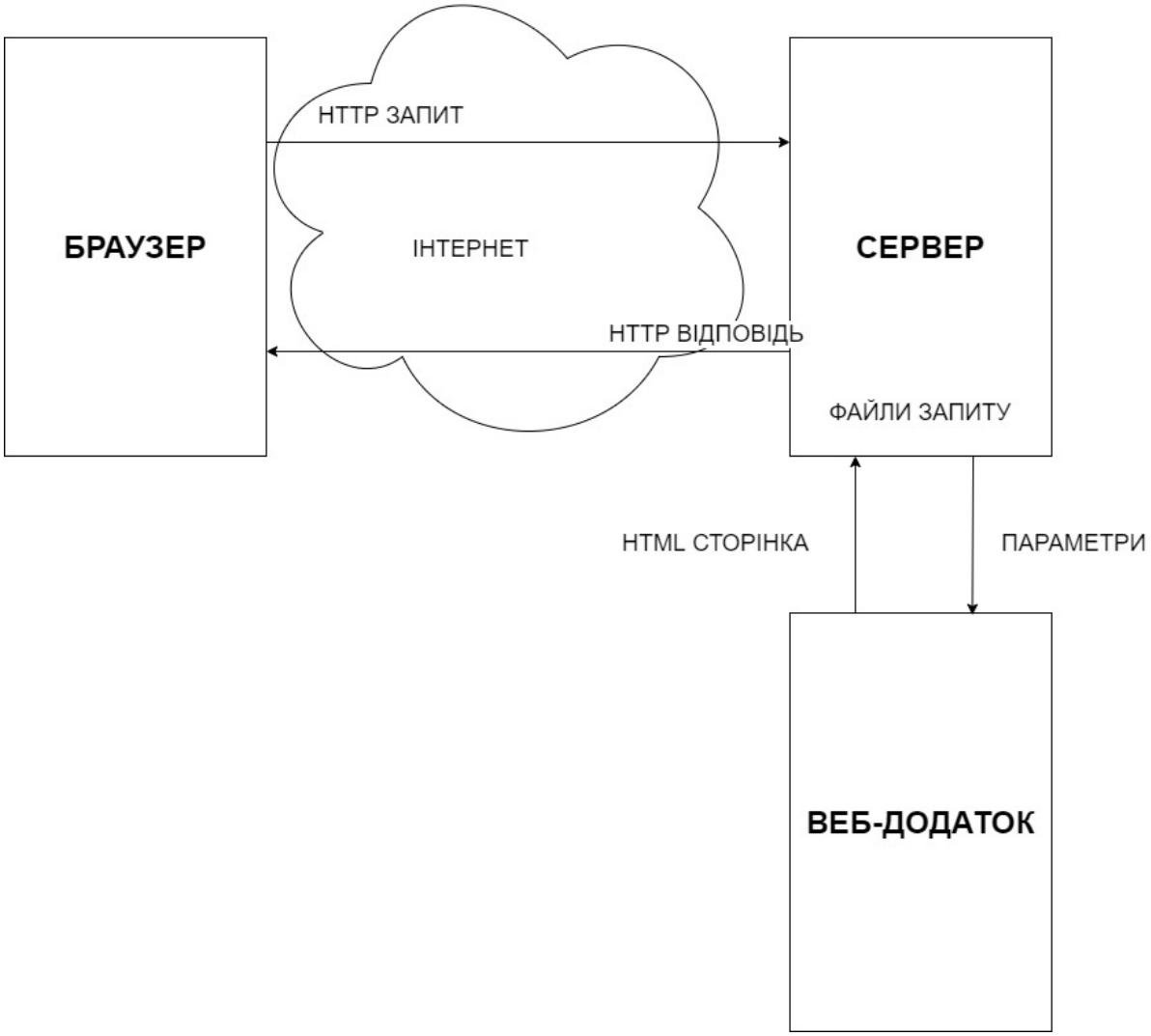


Рис. 1.3. – Схема роботи веб-додатку.

Складові частини веб-додатку[4]:

1.1. Клієнтська частина:

a. HTML та CSS - відображають інформацію у браузері;

b. JavaScript - змінює інформацію, яку відображаюсь HTML та CSS незалежно від оновлення сторінки;

1.2. Серверна частина:

a. скрипти і програми - створюють контент для клієнтської частини. До них звертаються за допомогою адрес з передачею необхідних параметрів.

b. база даних;

c. статичні файли;

1.3. Зовнішні сервіси та системи, з якими відбувається взаємодія.

Клієнтська частина (Frontend)[5][6] - це частина програми, яка реалізується на мовах HTML, CSS, JavaScript і запускається безпосередньо за допомогою браузера. Саме з цією частиною взаємодіє користувач до завершення сеансу роботи. Клієнтська частина охоплює розробку користувальницького інтерфейсу і розробку функціональності.

HTML - це мова розмітки, яка використовується для створення веб-сторінок. Ця мова визначає як, де і які елементи будуть відображатися на веб-сторінці. HTML визначає відображення таких елементів: зображення, відео, заголовки, параграфи, абзаци, посилання, списки, таблиці і т.д..

CSS - каскадні таблиці стилів, які відповідають за вид веб-сторінок в браузері. Стилі застосовуються до елементів сторінки для створення більш зручного інтерфейсу. CSS надає сайту насиченості завдяки можливості додати колір, тінь, фон, шрифт, обведення, анімацію і т.д.. Також, дозволяє скоротити HTML код та підвищити швидкість відкриття нових сторінок.

JavaScript - це мова програмування, яка використовується для надання веб-сторінці інтерактивності. Раніше ця мова застосовувалась виключно для клієнтської частини, але не так давно з'явився і серверний JavaScript. У клієнтської частини JavaScript використовується для створення динамічних скриптів на веб-сторінці. Він дозволяє управляти і анімувати елементи веб-сторінки.

Серверна частина (Backend)[7] відповідає за взаємодію з базою даних, продуктивність і бізнес-логіку. Перед тим, як відправити HTTP відповідь клієнту, в серверній частині запускається внутрішній код і генерується новий HTML-документ. Серверну частину представляють такі мови, як PHP, Java, .NET, Ruby, Python, NodeJS.

PHP – поширена мова програмування. Легко вбудовується в HTML-код і не вимагає інтегровану середу розробки. Має додаткові модулі для розробки веб-додатків.

Java, .NET – використовуються для створення підприємницьких додатків, коли потрібна реалізація всіх компонентів в єдиному середовищі.

Python - одна з популярних мов для веб-розробки. Її часто використовують для роботи з Big Data і великими об'ємними даними.

NodeJS - спеціалізований мова програмування, яка використовується для веб-додатків з великим числом одночасних підключень.

Переваги веб-додатку:

- Змінює вміст веб-сторінки. Як і динамічні сайти, веб-додаток може змінювати веб-сторінки. Сторінка формується після запиту клієнта з веб-сервера;
- Функціональність. Дозволяє зробити веб-сайт легким і доступним у використанні;
- Простота в оновлення та підтримки веб-сайту. Розробнику не потрібно змінювати кожну сторінку окремо, як в статичних сайтах;
- Створення складних сторінок. Складніші ніж в статичних та динамічних сайтах;
- Підключення до бази даних. Інформація зберігатися в базі даних, що дає можливість знайти та вилучити інформацію;

- Інтерактивність. Користувач стає активним учасником додатку. Це дозволяє зацікавити та утримати користувача на веб-сторінці;
- Простота доступу до додатку. На відміну від інших додатків, веб-додаток не вимагає завантаження. Користувач може зайти до додатку з будь-якого пристрою, в якому є доступ до Інтернету.

Недоліки веб-додатку:

- Повільне завантаження веб-сторінок. На відмінну від статичних та динамічних сайті, веб-додаток несе не тільки інформацію, а й інтерактивність. Через це веб-сторінки можуть завантажуватись повільніше;
- Вихідні файли сайту займають багато місця, тому можуть виникнути проблеми з оперативною пам'яттю сервера;
- Можуть виникнути проблеми з перенесенням на новий сервер;
- Дорожнеча. Дорожче ніж статичні та динамічні сайти.

Проаналізувавши роботу веб-додатків, можна стверджувати, що веб-додаток - це динамічний сайт, який крім інформаційної частини містить інтерактивну, яка дозволяє користувачу бути активним учасником сайту. Веб-додаток більш складніший і це збільшує вартість розробки.

1.2. Приклади додатків

BadParking – це український сервіс, який спрощує написання заяв в ДАІ. Сайт був розроблений для створення скарг з приводу псування шин транспортного засобу, неправильно припаркованого транспортного засобу. Щоб отримати заяву, необхідно заповнити форму, після чого на електрону пошту користувача надійде лист. Форма складається з:

- ПІБ.
- Телефон для зв’язку.
- Адреса для листування.
- Електрона пошта.
- Вид правопорушення.
- Номерні знаки транспортного засобу.
- Дата.
- Місто.
- Адреса події.

- Медіа файли. Можна прикріпити фото або додати посилання на відео.

Після заповнення форми, на електрону пошту, яку користувач вказав у формі, повинен прийти лист з заявою. Користувачу необхідно її роздрукувати та відправити на поштову адресу, яка вказана в листі. Головний недолік сервісу – користувач відправляє свої персональні данні невідомим особам, які потім ще формуються в документ та відправляються поштою.

Spot – це російський мобільний додаток, який був створений для спрощення подачі заяв з приводу порушення правил паркування або погіршення стану проїзної частини. Заява складається з:

- Фотографія.
- Номерні знаки транспортного засобу.
- Вид правопорушення.
- Коментар.
- Данні аккаунта: ПІБ, електрона пошта

Додаток одразу відправляє заяву в правоохоронні органи. Користувачі, також, можуть вносити деякі поправки в заяві та подивитися статус заяви. Розробники додали функцію, яка захищає фотографії від редагування і користувач не може відправити фото з Галереї. Додаток доступний для користувачів Android та iOS.

СтопПарк – це російський мобільний додаток, який доступний тільки для користувачів iOS. За допомогою додатку можна подати заяву за порушення правил дорожнього руху. Форма складається з:

- Адреса.
- Марка транспортного засобу.
- Номерні знаки транспортного засобу.
- Дата та час події.
- Данні аккаунта: ПІБ, електрона пошта.

- Фотографії. Додати можна мінімум 5 фотографій.
- Дата та час зробленого фото.
- Опис правопорушення.

Мобільний додаток складає з даних форми заяву та відправляє її в правоохоронні органи. Також, додаток дозволяє користувачу розглянути створенні заявки та перевірити їх статус.

Існує багато подібних проектів, які розроблені для того, щоб полегшити пересування людей та автомобілів. Але вони зв'язані лише з Правилами дорожнього руху. В Україні начисляється багато складних проблем на тротуарі, які загрожують безпеці руху пішоходів. Через цей привід було вирішено розробити веб-додаток для безпеки руху пішоходів.

1.3. Призначення веб-додатку для безпеки руху пішоходів

Тротуар є одним з головних елементів дороги. Він призначений для переміщення пішоходів і велосипедів. Однак, щодня пішоходи стикаються з багатьма проблемами на тротуарі. Найчастіше посеред дороги можна побачити рекламні споруди, припарковані транспортні засоби та рух транспортних засобів. Зазвичай малі архітектурні форми розміщуються на краю дороги та розмір їх конструкції скорочує встановлену нормами ширину пішохідного руху[8]. В Державно будівельних нормах України зазначається, що ширина однієї смуги пішохідного руху має бути кратною 0.75 м.. Всі ці елементи не тільки заважають пересування людей у пішохідній зоні, а й порушують закони України. У законах України вказана відповідальність за недотримання вимог будівельних норм і порушення правил дорожнього руху.

Щоб забезпечити безпеку, зручність і комфорт руху пішоходів, в Державно будівельних нормах України було додано пункт про тротуари, в якому зазначається, що не допускається встановлення рекламних конструкцій, тимчасових споруд. А тимчасові споруди повинні розташовуватися за межами тротуару або узбіч за вимогами чинних законодавчих та нормативних актів,

затверджених містобудівною документацією та місцевими правилами забудови населених пунктів. Адже незаконне розміщення тимчасових споруд може бути небезпечним не тільки для руху пішоходів, але й для життя пішоходів. Власники незаконних тимчасових споруд часто самовільно проводять електрику, через що щорічно згорають спорудження.

Водії транспортних засобів найчастіше припарковуються на тротуарах неправильно. В Правилах дорожнього руху[9] говориться, що на тротуарі стоянка заборонена, за винятком легкових автомобілів, які можуть бути поставлені на краю тротуарів, щоб рух пішоходів залишилося не менше 2 м., а також, якщо водій паркуєте в призначеному місці.

Ще однією небезпекою для пішоходів є рух транспортних засобів по тротуарах. За ПДР рух транспортних засобів по тротуарах допускаються в таких випадках:

- коли транспортний засіб виконує технологічні роботи;
- при відсутності інших під'їздів;
- коли транспортний засіб виконує обслуговування торгових чи інших підприємств;
- коли у водія є посвідчення водія.

Щоб уникнути цих порушення, був створений веб-додаток для забезпечення безпеки руху пішоходів. Веб-додаток спрощує подання та прийом заяв. Завдяки додатку українцям не треба буде дзвонити на гарячі лінії, стояти в черзі, писати заяву, роздруковувати заяву, а також відправляти її через пошту. Кожна людина може зареєструватися з будь-якого пристрою, вказавши своє ім’я, прізвище, номер мобільного телефону. Щоб подати заяву, їй необхідно вказати вулицю, обрати вид порушення та зробити дві та більше фото з місця події. Передбачається, що адміністратори додатку – це правоохоронні органи, які розглядають заяву, перевіряються місце.

1.4. Переваги та недоліки веб-додатку для забезпечення безпеки руху пішоходів

Як і усі додатки, веб-додаток для забезпечення безпеки руху пішоходів має свої переваги та недоліки[10].

Переваги додатку:

- Кожен користувач може відкрити веб-додаток з будь-якого пристрою, який має доступ до Інтернету;
- У всіх пристроях є системний браузер, тому користувачеві не потрібно встановлювати супутні програми;
- Сучасні браузери дозволяють створювати насичені інтерфейси, які вже добре знайомі користувачам. Тому користувачеві не потрібно проходити навчання користуванню додатком;
- Користувачі можуть отримати доступ до однієї і тієї ж версії веб-додатки, усуваючи будь-які проблеми сумісності;
- Користувачам не потрібно перевстановлювати програмні модулі на робочі станції;
- Користувачі не зберігають програмний продукт на своєму пристрої;
- Користувач може відновити пароль свого облікового запису;
- Спрощена система подачі заяв, які пов’язані з порушенням правил дорожнього руху та будівельних норм України. Громадянам не потрібно тратити час на дзвінки гарячим лініям, написання заяв, друк фото або заяви, та відправки заяв по пошті;
- Адміністратор може продивитися всі подані заявки;
- Користувачі можуть редагувати свої дані, а саме: ім’я, прізвище, номер телефон, поштова скринька, логін та пароль;
- Обліковий запис поділений на два види: користувач та адміністратор.

Недоліки додатку:

- Відсутність Інтернету. Так як веб-додатки знаходяться у всесвітній павутині, то без Інтернету доступ до веб-додатків заборонений. Якщо користувач захоче скористатися додатком через мобільний інтернет, то в нього можуть виникнути деякі складності через погану мережу;
- Дизайн можна змінити тільки з редагуванням програмного коду;
- Велика кількість користувачів в онлайні призводить до навантаження веб-сервера;
- Зловмисники можуть атакувати веб-додаток. Це може призвести до крадіжки даних користувачів і витоку всієї інформації додатки. Можлива втрата всієї інформації;
- Мало видів порушень.

1.5. Постановка задачі

Проаналізувавши предметну область, необхідно спроектувати і реалізовувати веб-додаток для забезпечення безпеки руху пішоходів. Додаток розділено на дві частини: клієнтська та серверна.

В клієнтській частині (frontend) реалізується користувальницький інтерфейс. В браузері користувача відображається інформація HTML і, а також формуються запити на сервер та обробляються.

В серверній частині (backend) реалізується робота зі сховищем даних, в якій зберігаються список користувачів, вхідні дані користувачів, а також відправлені заявки з даними від користувачів.

Веб-додаток повинен мати такі функції:

1. Реєстрація облікового запису. Користувачі повинні мати свою ідентифікацію, тому необхідно поставити вхідним даними користувачів обмеження:
   - Логін повинен бути унікальним;
   - Логін може містити тільки такий набір символів: . - _;
   - Пароль повинен відрізнятися від вже заданого логіна;

- Пароль повинен містити не менше 10 символів;
- Пароль повинен містити одночасно малі літери, великі літери, цифри і символи.

2. Вхід в обліковий запис. Необхідно задати перевірку наявності та збігу вхідних даних з базою даних PostgreSQL.
3. Створення заявки. У заявці користувач повинен ввести дані про порушення Правила дорожнього руху:
   - Не менше ніж дві фотографії з місця події, якщо це порушення Правил дорожнього руху, тоді в одній з фотографії повинно бути чітко видно номера транспортного засобу;
   - Обрати вид порушення;
   - Ввести адресу місця події;
   - Обрати місто, в якому трапилась подія;
4. Відправлення заявки. Всі дані повинні відправитися на сервер;
5. Зміна даних користувача;
6. Вихід з облікового запису. Для більшої безпеки необхідно надати користувачеві можливість вийти з системи.

Також, необхідно розробити шифрування та дешифрування даних, адже додаток повинен зашифрувати персональні дані користувача, щоб захистити їх від зловмисників.

ВИСНОВОК ДО 1 РОЗДІЛУ

Було проаналізовано статичні та динамічні веб-сайти, а також веб-додатки. В ході аналізу було вияснено, що на відміну від статичних та динамічних веб-сайтів, веб-додатки мають інтерактивність, яка залучує користувачів брати активну участь у роботі з додатком. Веб-додатком користувачі можуть користуватися з будь-якого пристрою, що робить його більш доступним та залучає більше користувачів. Було розглянуто багато додатків, які поліпшують пересування пішоходів, але переважно вони зв'язані з Правилами дорожнього руху. Порушити безпеку руху пішоходів можуть не тільки транспортні засоби, але й тимчасові споруди та рекламні конструкції. Це дуже актуальним в наш час, тому було вирішено розробити веб-додаток для безпеки руху пішоходів.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ

2.1. Архітектура додатку

Існують два підходи для створення об'єкта-орієнтованого веб-додатка: веб-форма та MVC шаблон.

Перший підхід заснований на наборі спеціальних веб-форм, які пов'язані з описами класів, об'єктів, які при виклику створюються та використовуються. Приклади: ASP.Net Web Forms, технологія JavaServer Faces.

Переваги:

- Наявність безліч готових компонентів;
- Швидкі для розробки веб-додатків;
- Добре працює з великими об'ємами даних, з якими працюють бізнес-додатки;
- Можна працювати без знань HTML, CSS та JavaScript.

Недоліки:

- Логіку призначеного для користувача інтерфейсу складно відокремити від коду;
- Складно використовувати кілька користувальницьких інтерфейсів в одному додатку;
- Кожна сторінка має свій контролер;
- Незручне використання автоматизованого тестування.

Другий підхід заснований на використанні наборів класів, відповідним шаблоном MVC. Це дозволяє отримати повний контроль над тим, що відбувається. Приклади: Java Spring, Java Struts, ASP.Net MVC.

Переваги:

- Повний контроль над HTML-кодом;

- Генерує чистий HTML-код;
- Розділяє користувацький інтерфейс та основну логіку веб-додатку;
- Легко використовуються JavaScripts та jQuery;
- Добре тестується;
- Добре інтегрується з Фреймворками.

Недоліки:

- Відсутній ViewState, стан вигляду;
- На відміну від веб-форм займає більше часу на розробку.

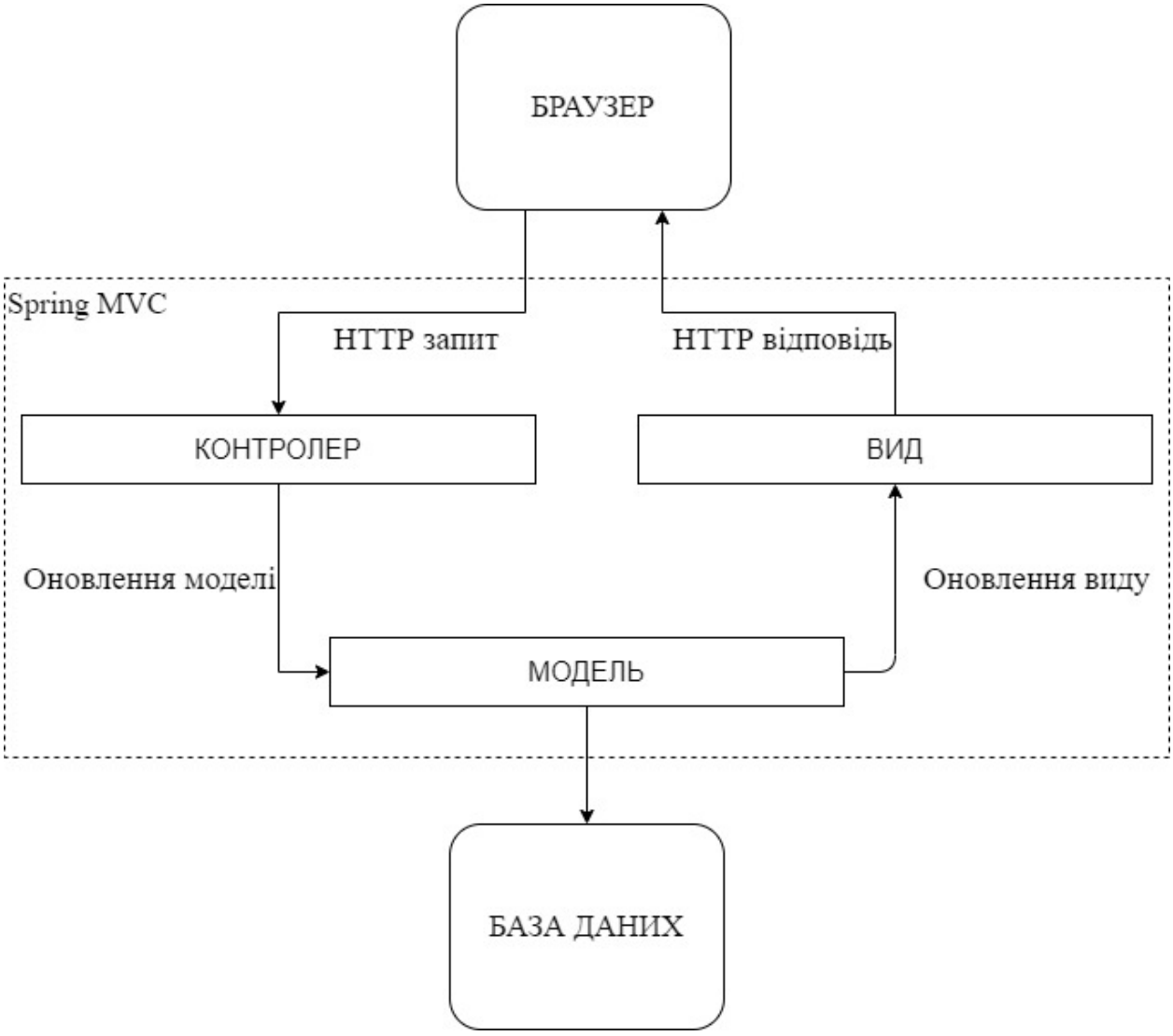


Рис. 2.1. – Схема роботи MVC Spring.

MVC розшифровується як Модель-Вид-Контролер. У цій архітектурі умовно поділені три компонента логічних частин програми:

1. Модель - це якийсь набір класів, які реалізують бізнес-логіку веб-додатку. Класи відповідають за управління даними: читання та запис в Базу даних, обробка даних. Також, взаємодіють між об'єктами, які становлять ці дані.
2. Форма - якийсь набір класів, які відповідають за відображення даних користувачеві в формованій HTML-сторінці або UI.
3. Контролер - відстежує дії користувача. Класи отримують якийсь запит даних до сервера і відправляють їх для поновлення Моделі. Після цього Контролер вибирає клас Виду для надання цих даних користувачеві. Контролер пов'язує Вид і Модель.

Завдяки поділу веб-додатку на компоненти, розробник може повністю контролювати формування HTML-сторінок. Це дозволяє повністю розділити логіки роботи веб-додатки, спростити структуру веб-додатки, полегшити тестування веб-додатки.

Шляхом поділу завдань ми отримуємо спрощене структурування веб-додатку. Однак, не варто забувати про його безпеку та продуктивність.

Для ефективної роботи веб-додатки необхідно:

- Використовувати багатошарову структуру. З її допомогою веб-додаток може складатися з декількох шарів, кожен з яких при зверненні до сусіднього шару може працювати як клієнт і сервер. Класична тривимірна архітектура складається з шарів: уявлення, бізнес-логіки та доступу даних. Це допоможе впорядковати програмний код, контролювати і оптимізувати продуктивність всіх шарів, та масштабувати додатки;
- Визначити стратегію взаємодії з іншими шарами. Так як програма має кілька шарів, необхідно пов'язати їх між собою;

- Використовувати кешування. Для підвищення продуктивності веб-додатку і швидкості відгуку призначеного для користувача інтерфейсу, необхідно використовувати механізм кешування. Цей механізм дозволить оптимізувати пошук даних і уникнути обходів мережі, а також зберегти результати довго стоячих обчислень, якими можна буде скористатися з кешу пізніше;
- Використовувати стратегію перевірки даних. Це допоможе захистити систему від декількох невдалих спроб ввести данні;
- Використовувати шифрування і дешифрування конфіденційних даних. Це допоможе уникнути передачі по мережі важливих даних у вигляді звичайного тексту;
- Обрати формат даних. Тобто, як будуть представлені формати даних;
- Обмежити доступ до файлових систем і ресурсів веб-додатки для безпеки даних веб-додатку.

Основні проблеми в розробці веб-додатків:

- Несанкціонований доступ;
- Зберігання важливих даних у вигляді звичайного тексту в базі даних;
- Неправильний розподіл ролей користувачів;
- Розкриття конфіденційної інформації кінцевому користувачу;
- Кешування конфіденційних даних;
- Відсутність перевірки авторизації користувача;
- Наявність складних і перевантажених сторінок;
- Наявність надмірних розмірів сторінок;
- Вибір невідповідного шаблону обробки запитів;
- Неправильне використання сховища.

Розглянемо області проектування веб-додатки:

1. Аутентифікація. Дозволяє провести перевірку автентичності зрівнюючи введений логін та пароль з базою даних користувача. Логін користувача

є головною функцією аутентифікації. Він дозволяє впізнати користувача. А головна функція пароля - безпечність інформації. Він запобігає від несанкціонованого доступу.

2. Авторизація. Після того, як користувач проходить аутентифікацію, цей механізм дозволяє визначити права доступу до додатка і ресурсів. Він з'ясовує які файли та інформацію необхідно надати користувачеві.
3. Кешування. Для підвищення продуктивності веб-додатку і швидкості відгуку призначеного для користувача інтерфейсу, необхідно використовувати механізм кешування. Цей механізм дозволить оптимізувати пошук даних і уникнути обходів мережі. При розробці служби кешування, необхідно врахувати шифрування конфіденційних даних перед кешуванням.
4. Управління винятками. Використовується для з'ясування помилки, яка може статися при роботі програми. Користувач повинен бути повідомлений про помилку додатка спеціальним повідомленням. Також, необхідно дозволити користувачеві повторити операцію. Однак, якщо помилка викликана збій системи або бази даних, необхідно записати її в спеціальний журнал. Також, варто уникнути розкриття конфіденційних даних через помилки.
5. Навігація. Цей механізм дозволить користувачеві легко і швидко переміщатися по сторінках. Щоб полегшити використання навігації, необхідно використовувати поширений шаблон навігацій, який вже знайомий користувачам: зробити посилання в одному стилі, а також задати стан посилань (активне посилання, посилання, на яку вказує курсор).
6. Перевірка введених даних. Перевірка даних є важливою ланкою в безпеці додатки. На стороні сервера і клієнта необхідно перевіряти всі вхідні дані, щоб уникнути помилок входу.
7. Шаблон. Щоб призначений для користувача інтерфейс виглядав охайним під час запуску програми з будь-якого пристрій, варто

розробити шаблон. Інтерфейс повинен бути простим і доступним у використанні.

2.2. Фреймворк Java Spring

Фреймворк[12] - це так званий набір бібліотек, які дозволяють зробити програму більш функціональною. У веб-розробці їх використовують для того, щоб надати динаміку додаткам. Фреймворки дозволяють створити певну архітектуру додатку та розділити контент веб-додатку від презентації. Частина, яка відповідає за контент включає за собою підтримку управління станом і сеансом, аутентифікація, відкриває доступ до даних. В основному фреймворки Java спрямовані на роботу з базою даних. За допомогою Spring можна створювати будь-які додатки на Java, в тому числі і веб-додатки. У свою чергу, фреймворк дозволяє створювати код з мінімальними витратами часу, так як він має безліч вбудованих інструментів.

Java Spring – це фреймворк з відкритим вихідним кодом для Java-платформи, який має власну MVC платформу. Фреймворк має безліч компонентів, які допомагають вирішити безліч проблем. Spring Security є компонентом Spring і надає механізми аутентифікації, авторизації, а також дозволяє убезпечити додаток.

Переваги Java Spring:

- Це полегшена платформа для створення веб-додатків;
- Можна створювати не тільки веб-додатки, а й додатків JEE (Java Enterprise Edition) ;
- Настройка компонентів відділена від програмного коду. Це полегшує керування проекту;
- Розділення архітектури на компоненти спрощує тестування додатку;
- Добра інтеграція з базами даних. Spring створює шар баз даних за допомогою Java Database Connectivity;
- Можливо автоматично зв’язувати компоненти;

- Немає необхідності створювати об’єкти вручну;
- Надає сервіс рівня middleware;
- Підтримує різні способи конфігурації.

Недоліки Java Spring:

- Складність в розробці;
- Складність в запуску.

Щоб зібрати Java-проект, необхідно використати Maven. Цей інструмент компілює проект, створює jar файл, створює дистрибутивні програми, генерацію документації.

Переваги Maven:

- Maven створює jar-файл і це дає можливість запустити його на будь-якій Операційній системі. Але для цього необхідно завантажити сторонні засоби;
- Можна з сервера скласти проект завдяки компіляції з командного рядка;
- Вирішує сторонні залежності. Складні проекти завжди використовують сторонні бібліотеки різних версій, Maven дозволяє вирішити конфлікти різноманітних версій і швидко переходить на нову бібліотеку;
- Можна відкривати усі Java проекти;
- Декларативне опис проекту.

Недоліки Maven:

- Залежить від встановленої версії Java;
- Працює повільніше ніж його аналаги.

За допомогою програми IntelliJ IDEA буде проведений процес складання проекту. У цій програмі можна буде створити Maven проект, після чого з'явиться файл pom.xml, який буде описувати дипломний проект. У цьому файлі можна буде додати використані бібліотеки проекту, після чого програма

запропонує завантажити бібліотеку в папку проекту. Однак, можна вибрати автоустановку бібліотек, так програма більше не буде давати запити з приводу установки бібліотек та стане встановлювати їх сама. Також, в файлі, який описує проект можна додавати не тільки назву бібліотек, але й вказати їх версію. Це зручно тим, що деякі версії бібліотек можуть конфліктувати між собою, та якщо виникне якийсь конфлікт, можна його залагодити зміною версії бібліотек. Також, крім описів використаних бібліотек pom.xml зберігає інформацію про збірку проекту. Можна додати інформацію про використання плагінів або ж вказати місце зберігання файлів проекту.

Оскільки дипломний проект використовує MVC шаблон, IntelliJ IDEA допоможе залишити серверну частину і в процесі запуску веб-додатку змінювати інтерфейс додатка. Перед запуском програми необхідно скомпілювати проект, запустити базу даних та сам додаток. В ході зміни шаблонів Freemaker додаток можна перекомпілювати, а потім перезапустити вкладку браузера.

Крім цього, IntelliJ IDEA дуже допомагає розробнику при написанні коду. Програма підтримує синтаксис багатьох типів файлів, тим самим підсвічує синтаксис, а також допомагає оформити код в одному стилі і створити лаконічну структуру проекту. Програма має безліч зручних функцій, які спрощують роботу з кодом, наприклад, можна натиснути на ім'я функції і програма відразу ж відправить розробника в місце будь-якого файлу, в якому викликається ця функція. Хочеться відзначити, що програма має платну і безкоштовну версію. Безкоштовна версія включає всі функції, які потрібні для створення веб-додатку, проте, в ній відсутнє відображення проекту у вигляді UML діаграм. Для того, щоб описати класи проекту, в майбутньому, я скористаюся платної версією програми, якою можна користуватися безкоштовно протягом 30 днів.

Написання коду - це важливе завданням кожного програміста, проте для того, щоб скоротити час розробки програми, необхідно спроектувати

програму. Проектування додатка являє собою певну документацію, в якій розташовуються різні діаграми, текстові описи і модель майбутньої програми. Програміст ставить цілі і фокусується на спроектованому проекті. Це дозволяє йому не відволікатися на інші завдання і запобігти безлічі помилок. UML - один з найпопулярніших мов проектування. Ця мова була розроблена для аналізу та проектування проектів. Вона складається з набору символів, які мають деяке значення та правилами маніпулювання ними.

Переваги UML[13]:

- Допомагає проаналізувати і спроектувати проект;
- Легка у вивченні синтаксису;
- Дозволяє краще зрозуміти суть завдання і спосіб її реалізації;
- Можна розглянути задачі з усіх точок зору.

Недоліки UML:

- Необхідно вивчити синтаксис мови;
- Застосовується до досить вузьким завданням: проектування графічного інтерфейсу, формуванню структури БД.

Елементи проектування UML складаються з сутностей, діаграм і класів.

Сутність - це якась абстракція, за допомогою якого створюються моделі. Вони бувають різних видів:

- Структурні. Являються іменниками. Основним видом таких сутностей в діаграмах класах є клас;
- Поведінкові. Відповідають за динамічну частину. Вони відповідають за подання поведінки моделі, описуючи якісь дії або події системи;
- Груповані. Це блоки, на які можна розкласти модель. Сутності організовують різні компоненти об'єктної моделі;
- Анотація. Це пояснювальні частини UML-моделі. Якийсь коментар, який виноситься для опису елементів моделей.

Діаграма - це граф, у якого замість вершин сутності, а замість ребр відносини. UML використовуються наступні діаграми:

- Діаграма варіантів використання;
- Діаграма класів;
- Діаграма станів;
- Діаграма діяльності;
- Діаграма розгортання;
- Діаграма послідовності;
- Діаграма кооперації;
- Діаграма компонентів;

Перелік цих діаграм і їх назви є канонічними в тому сенсі, що представляють собою невід'ємну частину графічної нотації мови UML. Більш того, процес побудови моделі нерозривно пов'язаний з процесом побудови цих діаграм. При цьому сукупність побудованих таким чином діаграм є самодостатньою в тому сенсі, що в них міститься вся інформація, яка необхідна для реалізації проекту складної системи.

Усі ці діаграми зображують і описують різні уявлення про складові системи в мові UML. Оскільки діаграма варіантів використання застосовується як загальна продумана модель складної системи, яка є основною для проектування решти діаграм. Можна сказати, що діаграма класів – це деяка логічна модель або логічне проектування, яка має статичне та динамічне представлення. Тобто, модель описує такі елементи, як функціональність проекту, класи проекту та взаємодію об’єктів між собою. Також, стан цих об’єктів.

2.3. Діаграма варіантів використання

Візуальне моделювання в UML можна уявити як певний процес спуску по рівням від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної

системи. Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування. Діаграма варіантів використання є вихідним концептуальним поданням або концептуальною моделлю системи в процесі її проектування і розробки.

Розробка діаграми варіантів використання переслідує мети:

- Визначити загальні межі і контекст модельованої предметної області на початкових етапах проектування системи.
- Сформулювати загальні вимоги до функціонального поведінки проектованої системи.
- Розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.
- Підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми складається в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на модельовану систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою[13]. Діаграма системи приведена на рис.2.2.

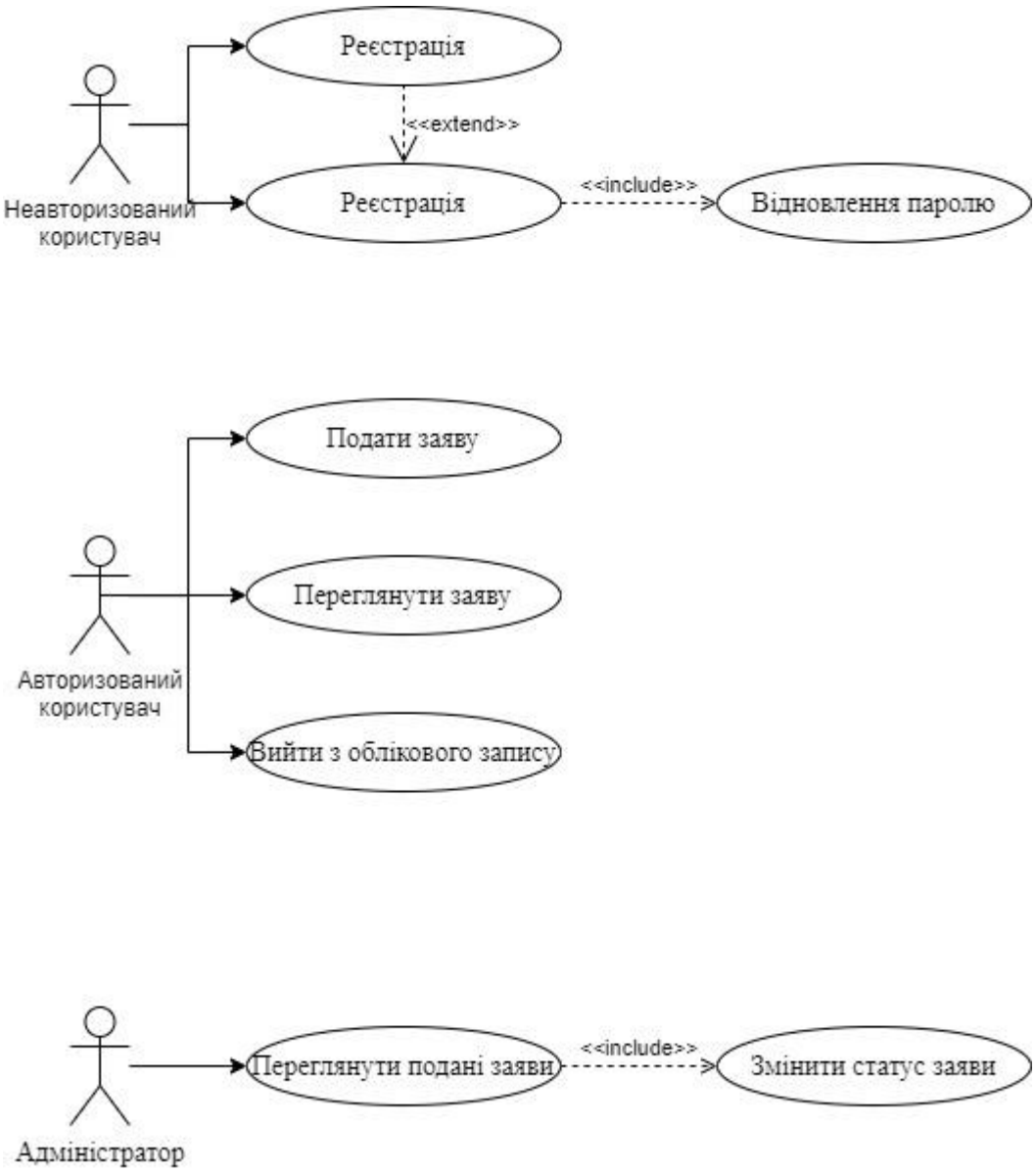


Рис. 2.2. – Діаграма варіантів використання (Use Case diagram).

Головними персонажами додатку являються користувач і адміністратор. Користувач розділені на авторизовані і неавторизованих.

Неавторизовані користувачі можуть авторизуватися в додатку, а також зареєструватися.

Авторизований користувач може додати заявку, вийти з особистого кабінету та відредагувати дані профіля. Він може змінити ім’я, прізвище, номер телефону, e-mail, логін та пароль.

В свою чергу Адміністратор може переглядати піддані заявив, а також використовувати пошук заяв по адресам.

2.4. Функціональність додатку

Для реалізації аутентифікації необхідно використати Spring Security. Коли користувач вирішить зайти в систему, додаток повинен буде надати користувачеві форму. В формі користувач вводить свій логін та пароль після чого завдяки Spring Security можна буде перевірити існує заданий логін чи ні, та підходить до нього заданий пароль. Якщо ні, то додаток повинен видати помилку

Авторизація з використанням Spring Security.

Після успішної аутентифікації додаток визначає роль користувача. Чи є в нього права адміністратора чи ні. Згідно з роллю користувача, надається певна сторінка. Якщо користувач має права адміністратора, він отримує доступ до сторінки за заявками. Користувач може переглянути усі заявки, а також зробити пошук заяв по адресі. Якщо користувач не має прав адміністратора, йому надається сторінка, в якій він може подати заявку.

Реєстрація

Для реєстрації необхідно буде створити контролер, який відповідає за реєстрацію. Він повинен приймати ім’я та тип компоненту.

Редагування профілю

Можна створити контролер «Користувач», який буде не тільки приймати інформацію о користувачеві після реєстрації, а також відновлювати дані.

Подача заяв

Необхідно створити головний контролер, який буде приймати інформацію користувача, зберігати її до бази даних, та у випадку, якщо користувач має права адміністратора, необхідно відобразити усі подані заявки.

Пошук заяв

Якщо користувач має права адміністратора, йому повинні відображатися усі подані заявки. Щоб адміністратору було легше переглядати її, необхідно розробити пошук заявок по адресі. Адміністратор повинен вводити в певний блок назву вулицю, та додаток повинен сприймати введені дані як інформацію без розділових знаків та інформацію, яка не включає великі літери.

Проектування баз даних

База даних є програмним забезпеченням на сервері. Вона розміщена на сервері і серверна частина веб-додатка звертається до неї для отримання даних, які потрібні для формування HTML-сторінки. Її головна роль - зберігати дані і за запитом видавати їх. У базі даних може зберігатися абсолютно будь-яка інформація, будь-то інформація користувача, коментарі або пости. PostgreSQL отримала велику популярність завдяки легкості, масштабованість і продуктивністю. PostgreSQL[15] – це система управління базами даних з відкритим кодом. Вона працює на багатьох платформах ОС: Linux, Windows та Mac. Для того, щоб створити таблиці, необхідно завантажити її.

Для початку треба проаналізувати сутності бази даних та спроектувати базу даних. Це дозволить нам уникнути дублікатів, розробити базу даних таким чином, щоб сервер отримував саме запрошувані дані. Спочатку оберемо сутності для бази даних:

- Користувач.
- Посилання.
- Роль.

Для кожної сутності створимо таблицю, в яку входить: назва атрибута, опис атрибута, та тип даних. Це дозволить в подальшому ході роботи спростити створення бази даних.

Перш за все первинні ключі повинні містити унікальні значення. Таблиця може мати лише один такий ключ та ім’я ключа не повинно збігатися з іменами інших атрибутів. Як бачимо в таблиці 2.1. атрибут «ID» виконує роль ідентифікатора таблиці «Користувач». В цій таблиці зберігається інформація з реєстрації користувача: Ім’я, Прізвище, Дата народження, електрона скринька та номер телефону.

Табл. 2.1. – Структура таблиці «Користувач».

Атрибут	Опис атрибута
ID	Первинний ключ
ACTIVE	Авторизація
FIRST_NAME	Ім’я користувача
LAST_NAME	Прізвище користувача
EMAIL	Електрона пошта
PHONE_NUMBER	Номер телефону
USERNAME	Логін користувача
PASSWORD	Пароль

В таблиці 2.2. зберігається інформація про відправлені заявки, а саме: логін користувача, коментар, вид порушення, місто, адреса та коментар.

Табл. 2.2. – Структура таблиці «Адміністратор».

Атрибут	Опис атрибута
ID	Первинний ключ
FILENAME	Ім’я фото
COMM	Коментар
TYPE	Вид порушення
CITY	Місто
ADR	Адреса
USER_ID	Зовнішній ключ

В таблиці 2.3. зберігається інформація про роль, яку користувач отримує при авторизації. Зовнішній ключ забезпечує зв’язок з іншою таблицею. В даному випадку з «ID».

Табл. 2.3. – Структура таблиці «Роль»

Атрибут	Опис атрибута
USER_ID	Зовнішній ключ
ROLES	Роль

Проаналізувавши усі таблиці, можна зробити загальну схему баз даних.

```
create table form (
    id int8 not null,
    filename varchar(255),
    comm varchar(255),
    type varchar(255),
    city varchar(255),
    adr varchar(2048) not null,
    user_id int8,
    primary key (id)
);
```

Рис. 3.2. – Опис таблиці form.

```
create table user_role (
    user_id int8 not null,
    roles varchar(255)
);
```

Рис. 3.3. – Опис таблиці user_role.

```
create table usr (
    id int8 not null,
    active boolean not null,
    mail varchar(255),
    number varchar(255),
    first_name varchar(255),
    last_name varchar(255),
    password varchar(255) not null,
    username varchar(255) not null,
    primary key (id)
);
```

Рис. 2.3. – Опис таблиці usr.

```
alter table if exists form
    add constraint form_user_fk
    foreign key (user_id) references usr;

alter table if exists user_role
    add constraint user_role_user_fk
    foreign key (user_id) references usr;
```

Рис. 2.4. – Опис зв'язку таблиці form та user_role.

Проаналізувавши усі таблиці, можна зробити загальну схему сутностей (рис.2.5).

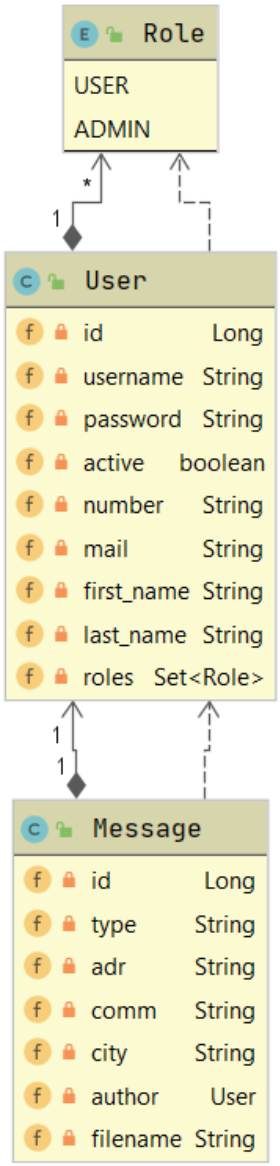


Рис. 2.5. – Схема сутностей.

ВИСНОВОК ДО 2 РОЗДІЛУ

Проаналізувавши архітектуру веб-додатку, було вирішено обрати фреймворк Java Spring, завдяки якому можна розробити архітектуру MVC. Було описано варіанти використання веб-додатків завдяки діаграмам UML, а також розглянуто Spring Security, який допоможе розробити аутентифікацію та авторизацію веб-додатку, а Maven закопилює додаток. Також в даному розділі дипломного проекту була описана база даних проекту. Розглянувши сутності окремо – дозволило встановити точні зв’язки між сутностями, що в подальшому ході розробки проекту полегшило роботу.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ

3.1. Реалізація веб-додатку

За допомогою програми IntelliJ IDEA створюємо Java проект з використанням Maven. Оскільки кожен проект має назву, було вирішено назвати проект JustGO. Щоб продовжити подальшу роботу, необхідно в файлі pom.xml підключитися до Spting Framework, Postgresql, Flywaydb.

У другому розділі було спроектовано базу даних. Для того, щоб реалізувати її необхідно включити pgAdmin і створити базу даних. База даних буде називатися justgo і для майбутнього тестування я задам їй пароль "123" та власника «root». Всі ці дані необхідно вказати у файлі проекту application.properties. Цей файл призначений для налаштування та зміни функціоналу веб-додатку. Необхідно вказати url дати бази, ім'я користувача і пароль. Оскільки в додатку користувач розділені на ролі, необхідно створити міграції Бази Даних. Вони розміщені в папці db/migration. Згідно з документацією назву файлів необхідно підтримуватися таких файлів:

1. *Файл* починається з англійської букви V.
2. Далі йде номер версії міграції.
3. Дворазове підкреслення.
4. І опис міграції.

У проекті програми розміщено два файли: V1__Init_DB.sql і V2__Add_Admin.sql.

У першому файлі створимо таблиці користувача, роль користувача і форми, а в другому додаємо адміністраторів додатки.

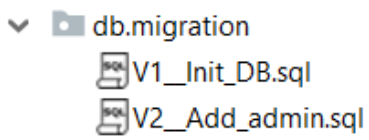


Рис. 3.1. – Папка db/migration.

Основа папка Backend - java/com/example/justgo. У ній я створюю папку domain, в якій будуть описані всі сутності: Message, Role, User. У кожній сутності задані атрибути в доступі приват, що дозволяє обмежити доступ до даних. Атрибути видно в одному класі, однак getter і setter дозволять контролювати цими даними. Оскільки заявка відправляється адміністратору з аргументами суті User: ім'ям, прізвищем, логіном та номером телефону користувача, необхідно пов'язати сутність Form і User. Для цього я використовую зв'язок ManyToOne. Також, необхідно додати зв'язок User і Role.

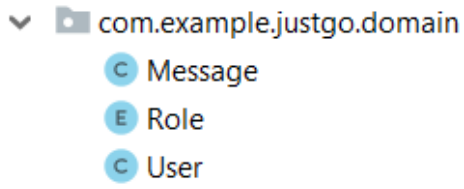


Рис. 3.2. – Папка domain.

Щоб додати і отримати інформацію з бази даних, я створюю папку controller. Папка містить контролери, які відповідають за обробку даних заяв, оновлення інформації користувачів і реєстрації. У кожному контролері присутні анотації GetMapping і PostMapping. GetMapping використовується для методу типу get, який буде обробляти http-запит, а PostMapping для методу post, який також обробляє http-запит.

FormController відповідає за головну сторінку веб-додатку, з якою може взаємодіяти авторизований користувач. Тут присутні дві анотації GetMapping, оскільки, якщо користувач не авторизований, він відправляється на головну сторінку веб-додатка, в якій він може обрати наступні дії: авторизуватися або ж зареєструватися. Якщо користувач авторизувався і він є адміністратором, то він може переглянути всі відправлені форми. А що щодо звичайного користувача, він може заповнити заявку і отримані дані відправляться в базу даних. Оскільки фотографії зберігаються на сервері, було вирішено зашифрувати їх назва, щоб убезпечити доступ до них.

Контролер, який відповідає за реєстрацію користувача пов'язаний з класом UserService. Цей клас обробляє дані користувача. Саме він обробляє логіни користувачів і визначає зареєстрований користувач чи ні. Якщо користувача з заданим логіном не існує, тоді він успішно проходить реєстрацію і його дані заносяться в базу даних. Однак, якщо користувач з таким іменем вже існує в базі даних, тоді контролер видасть незареєстрованому користувачеві помилку.

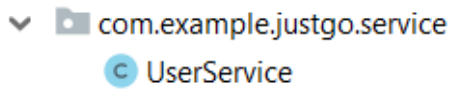


Рис. 3.3. – Папка service.

Призначений для користувача контролер відповідає за оновлення інформації користувача. Оскільки людина може змінити своє ім'я, прізвище, номер телефону – додаток надає користувачеві можливість зробити це і в своєму профілі. І звичайно ж, змінити пароль і логін. У разі зміни логіна, знову викликається метод для перевірки існування користувача в базі даних.

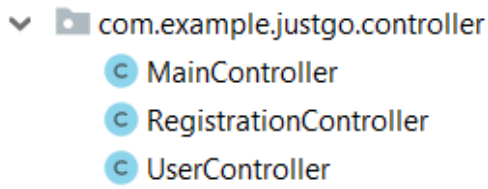


Рис. 3.4. – Папка controller.

Папка config відповідає за конфігурацію. Конфігурація MVC реалізує MVC шаблон. Вона дозволяє отримати повний доступ над додатком і розділяє його на три компоненти: модель, форма і контролер. Тобто, ми отримуємо:

- доступ до класів, які відповідають за управління даними;
- доступ до відстеження дій користувача;
- доступ до класів, які відображають дані в форматі HTML.

WebSecurityConfig відповідає за налаштування безпеки. Саме тут відбувається перевірка: авторизований користувач чи ні. Також, тут описуються методи, які переводять користувача. Якщо користувач вийшов з системи, то його

переводять головну сторінку. Якщо користувач зареєструвався, то його переводять на сторінку входу в додаток.

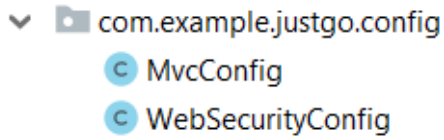


Рис. 3.5. – Папка config.

Клас Application використовується для запуску веб-додатку. Однак, перед тип як запустити додаток, необхідно запустити базу даних.

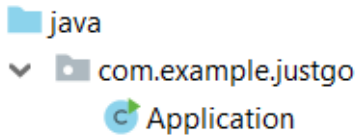


Рис. 3.6. – Папка justgo.

Основна папка Fronted - resources. Було вирішено розділити її на дві папки: templates і assest. У assets зберігаються зображення і стиль, які використовуються в HTML-сторінках.

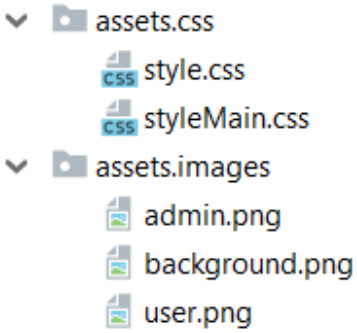


Рис. 3.7. – Папка assets

У templates зберігаються шаблони FreeMarker. FreeMarker дозволяє вставити дані в HTML-документ. Він може використовувати макроси, списки, умови і не тільки. В папці details описані макроси, які використовуються в інших документах. Оскільки HTML-сторінки мають однакову конструкцію, щоб не переписувати код, було створено два шаблони з макросами: commonIn

і commonOut. commonOut - це структура сторінок, які відображаються для незареєстрованих користувачів, наприклад: головна сторінка, вхід в особистий кабінет, реєстрація. У свою чергу commonIn використовується для авторизованого користувачів. Він розділяє профіль користувача від заявок.

Щоб користувачеві було зручно заповнювати заявку, було вирішено додати підказку в містах. Користувач може вибрати місто зі списку або ж почати вписувати його, після чого підказка підлаштовується під введені дані так користувач може зекономити час. Оскільки в Україні число міст складає 460, було вирішено зробити окремий шаблон з макросом city, щоб відокремити форму.

Макрос шаблону login зберігає форми. Оскільки в реєстрації і авторизації користувач вводить однакову форму - логін і пароль, за допомогою умови можна вивести ці форми в дві сторінки, а для реєстрації залишається введення імені, прізвища, номери телефону і e-mail. Також, в цьому шаблоні описаний макрос виходу з системи.

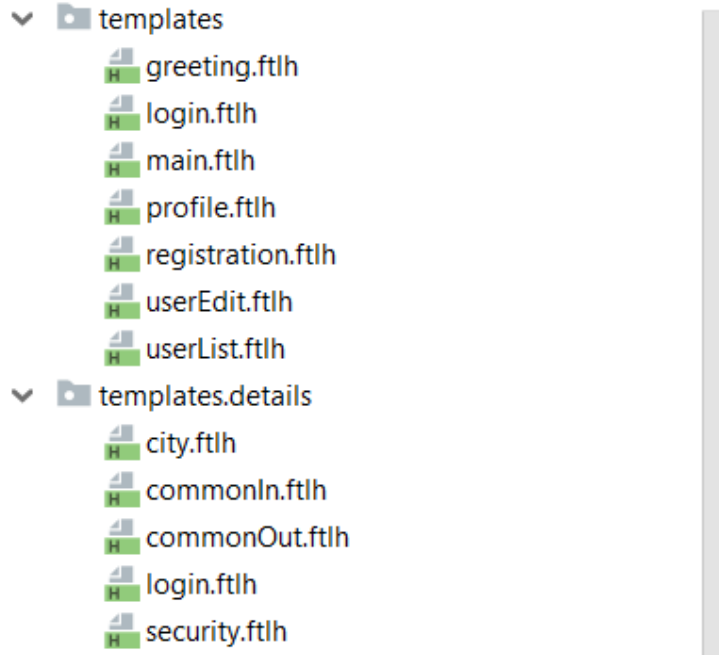


Рис. 3.8. – Папка templates

3.2. Робота веб-додатку

Щоб запустити веб-додаток, необхідно включити базу даних, закампілювати проект та запустити клас Application. Для тестування веб-додатку необхідно перейти по посиланню localhost:8080/.

При переході за посиланням нас зустрічає головна сторінка веб-додатку (рис. 3.9). Комфортний простий інтерфейс зі зручною навігацією пропонує користувачеві зайти в особистий кабінет або ж зареєструватися в додатку.

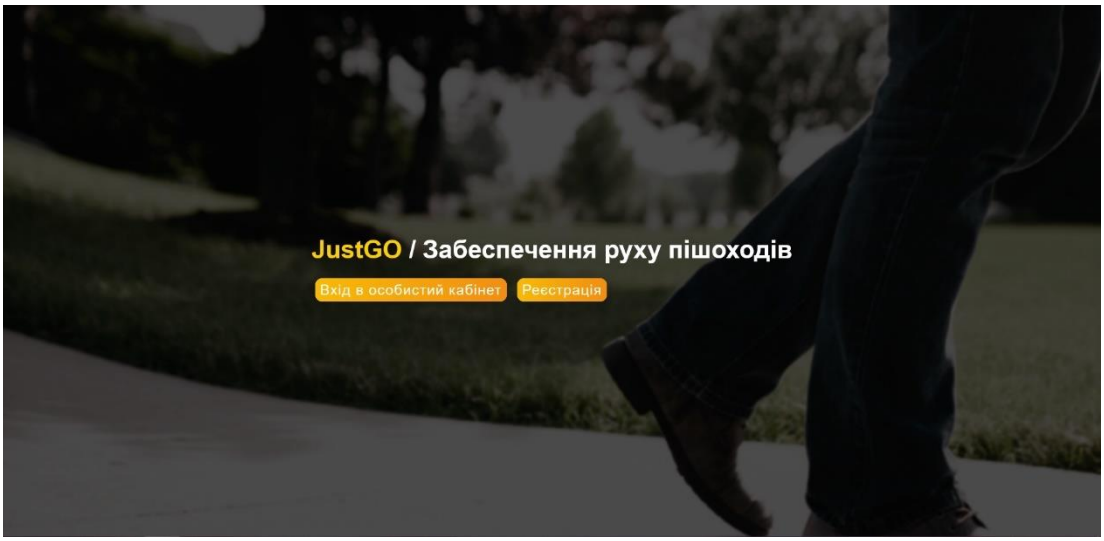


Рис. 3.9. – Головна сторінка веб-додатку

Якщо користувач вирішив зареєструватися в веб-додатку, додаток попросить користувача ввести логін, пароль, ім'я, прізвище, номер телефону/ e-mail (рис.3.10).

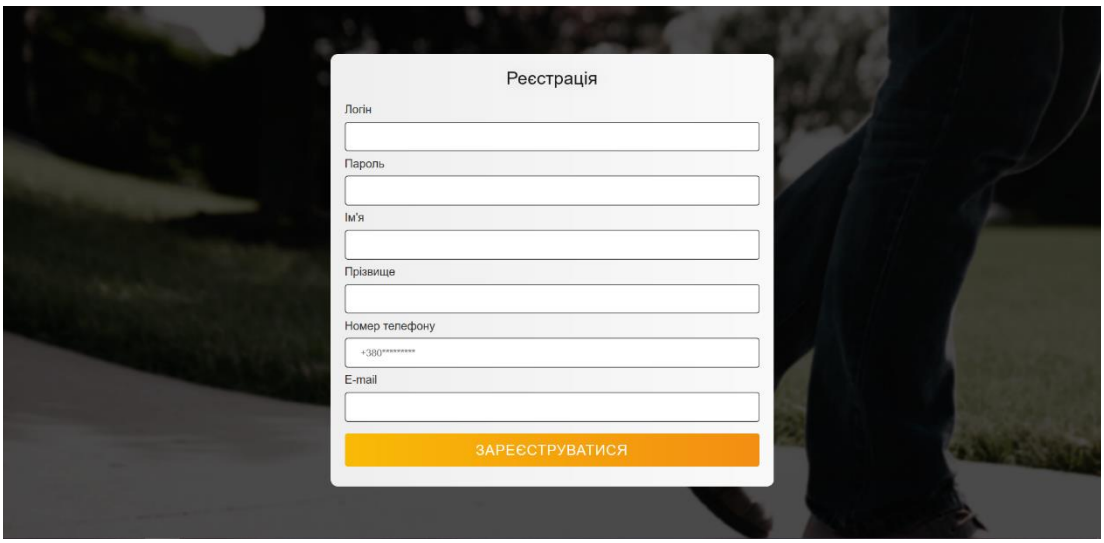


Рис. 3.10. – Сторінка реєстрації веб-додатку.

Якщо користувач ввів логін, який вже зайнятий іншим користувачем, веб-додаток відображає помилку (рис. 3.11).

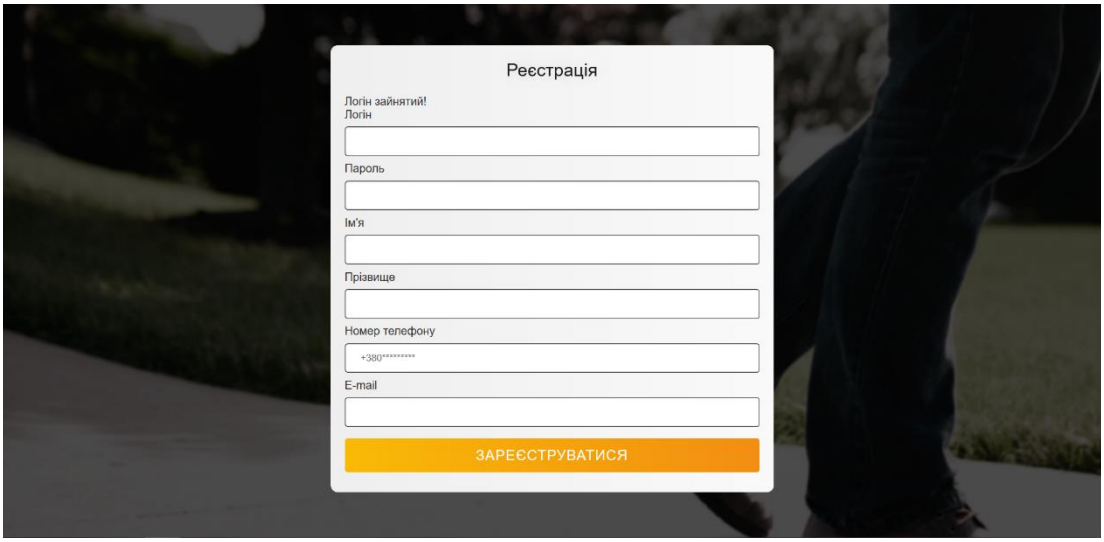


Рис. 3.11. – Помилка в реєстрації веб-додатку, зайнятий логін.

Оскільки кожний e-mail має певну структуру, а саме налічує равлика(@) та крапку(.). Також, він не має складатися з пробілу, знаку питання, знак оклику і т.д.. Якщо користувач ввів неправильно свою поштову скриньку, веб-додаток відобразить помилку (рис. 3.12).

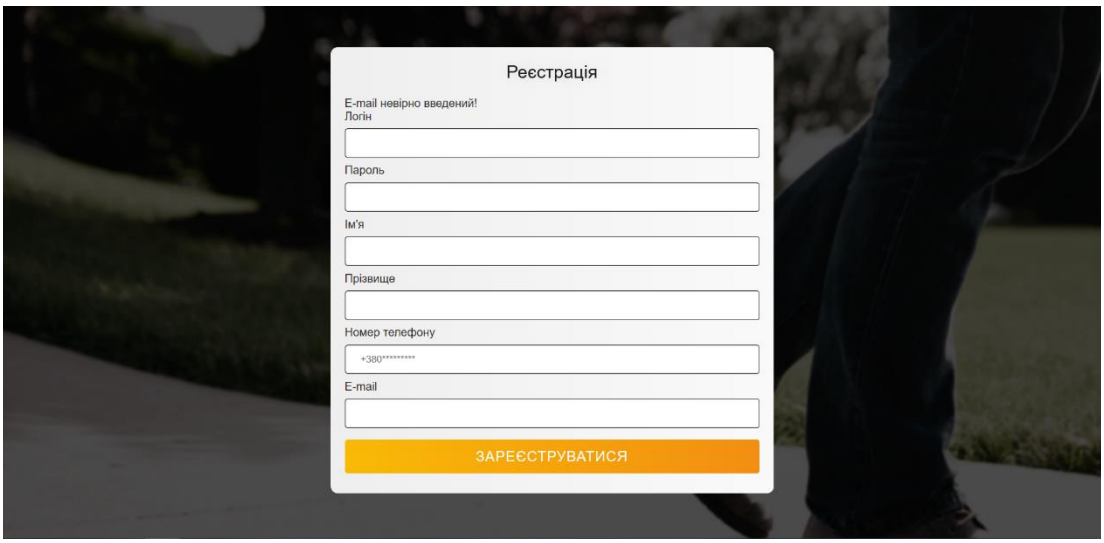


Рис. 3.12. – Помилка в реєстрації веб-додатку, невірний e-mail.

Як і e-mail, номер телефон також має свою структуру. Він має складатися з 11 цифр. Якщо користувач ввів номер телефону неправильно, веб-додаток відображає йому помилку (рис. 3.13).

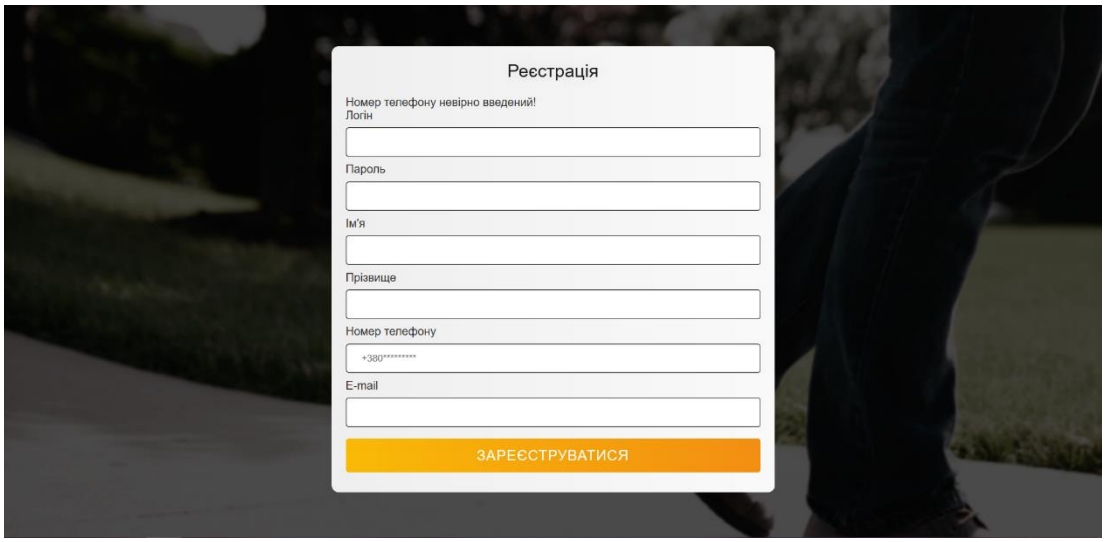


Рис. 3.13. – Помилка в реєстрації веб-додатку, невірний номер телефону.

Після того, як користувач зареєструвався, веб-додаток відправляє його до сторінки з авторизацією (рис. 3.14).

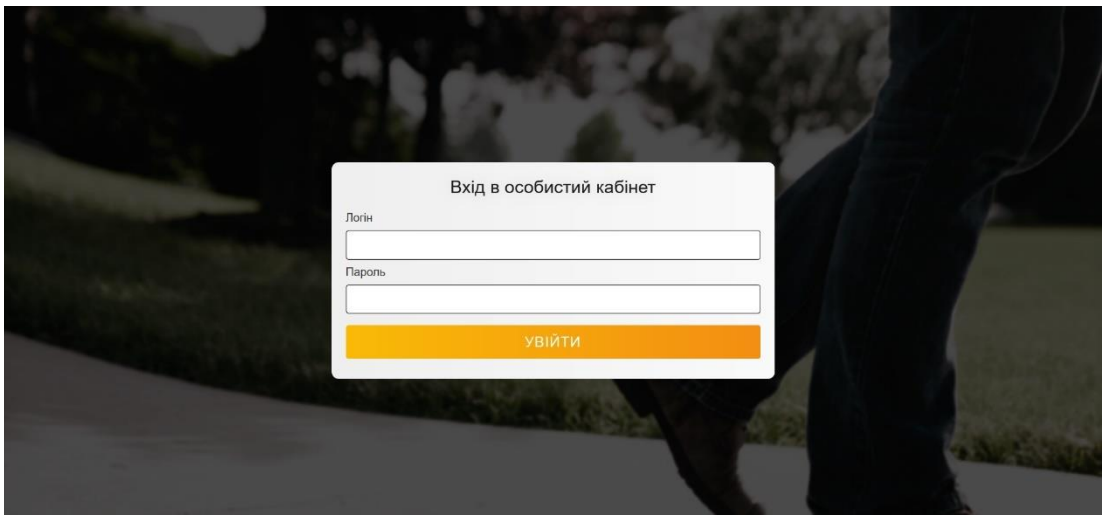


Рис. 3.14. – Вхід в особистий кабінет.

Якщо користувач ввів невідні данні, веб-додаток відображає йому помилку (рис. 3.15)

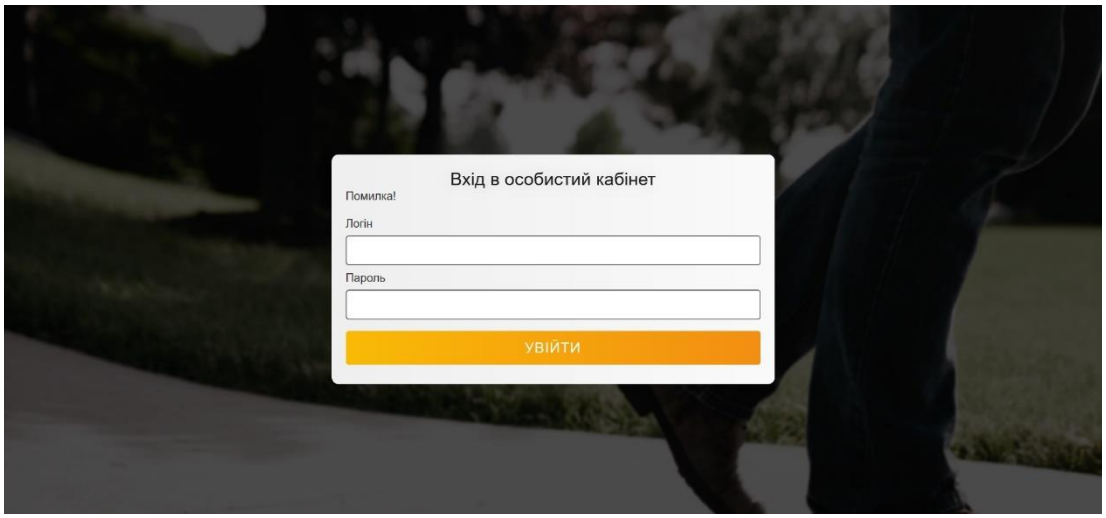


Рис. 3.15. – Помилка у вході в особистий кабінет.

Коли користувач ввійшов в особистий кабінет, веб-додаток відправляє його на сторінку main, яка відображається для кожного користувача по різному. Якщо користувач не має прав адміністратора, веб-додаток пропонує йому заповнити заявку та ввести місто, адресу, вид порушення, коментар та додати картинку (рис. 3.15).

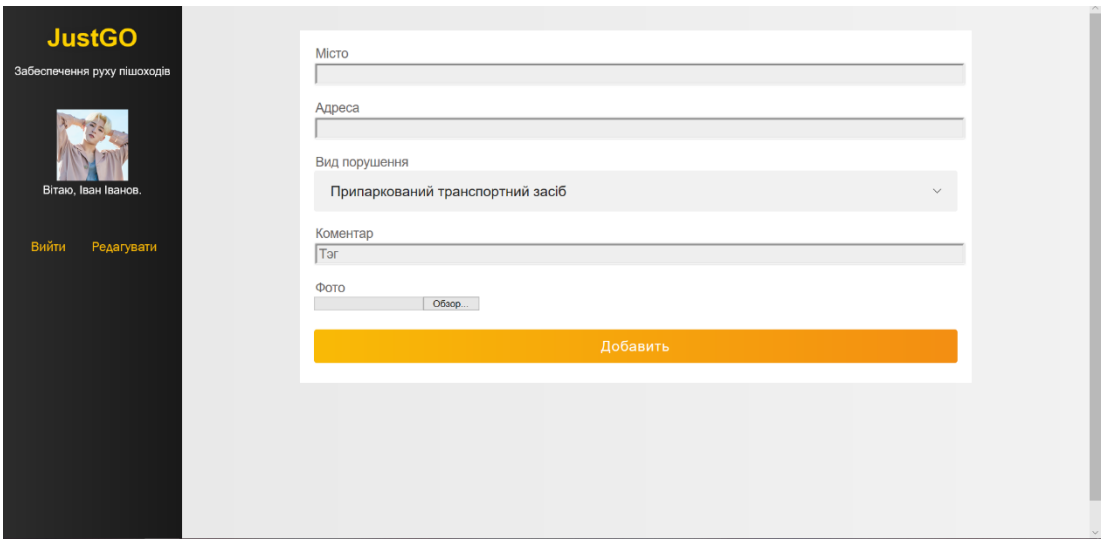


Рис. 3.16. – Сторінка користувача.

Коли користувач вводить назву міста, додаток відображає список з усіма містами України (рис. 3.17).

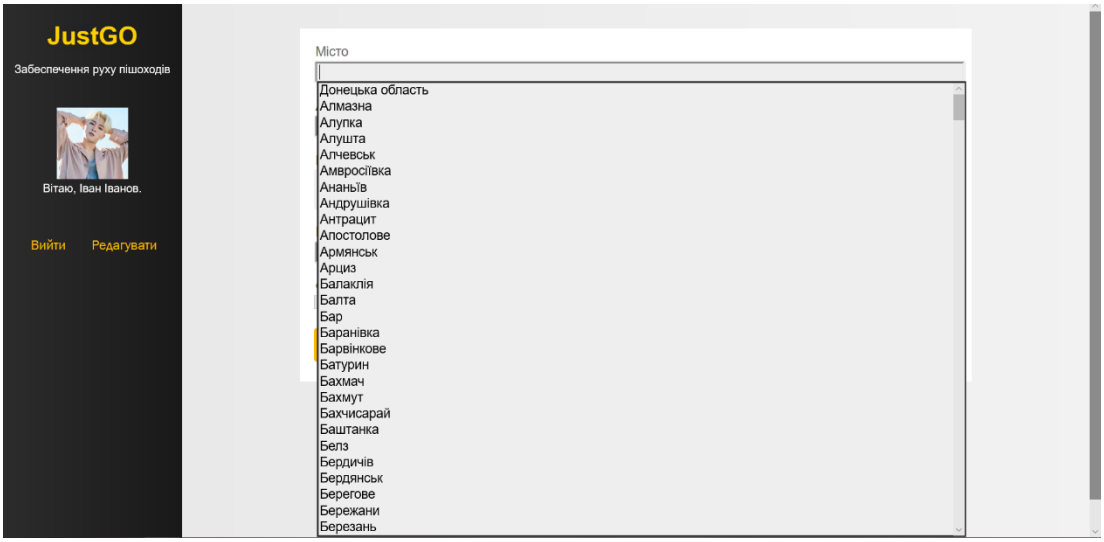


Рис. 3.17. – Список міст.

Коли користувач починає вводити назву міста, веб-додаток налаштовується до заданих даних та рекомендує користувачеві найліпші варіанти (рис 3.18).

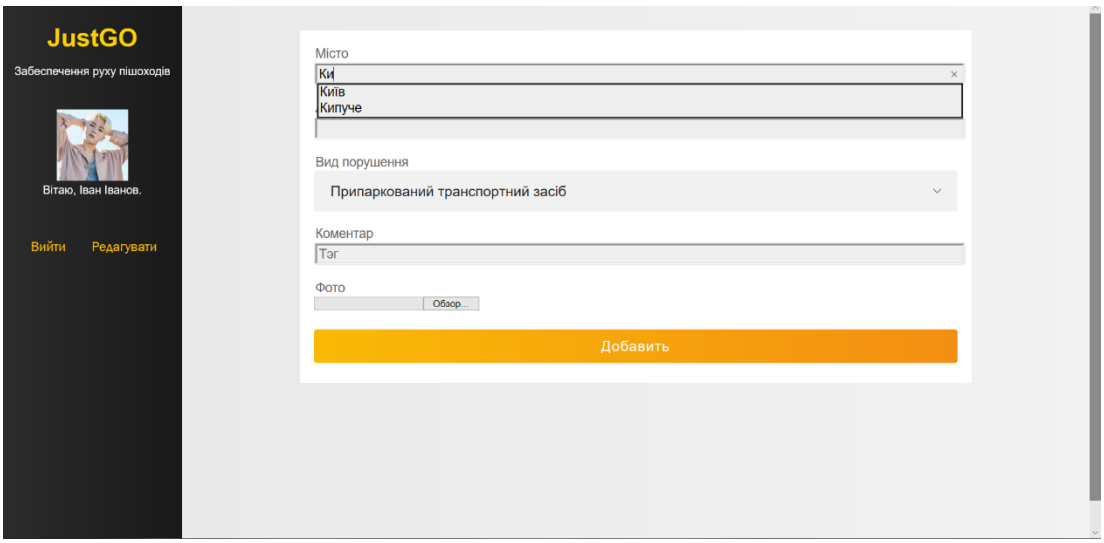


Рис. 3.18. – Найліпші варіанти міст.

Веб-додаток розрахований на певні види порушення, тому він сам пропонує обрати вид порушення:

- *Припаркований* транспортний засіб;
- Рухомий транспортний засіб;
- Рекламна споруда;
- Тимчасова спору*да.*

Детальніше в рис. 3.19.

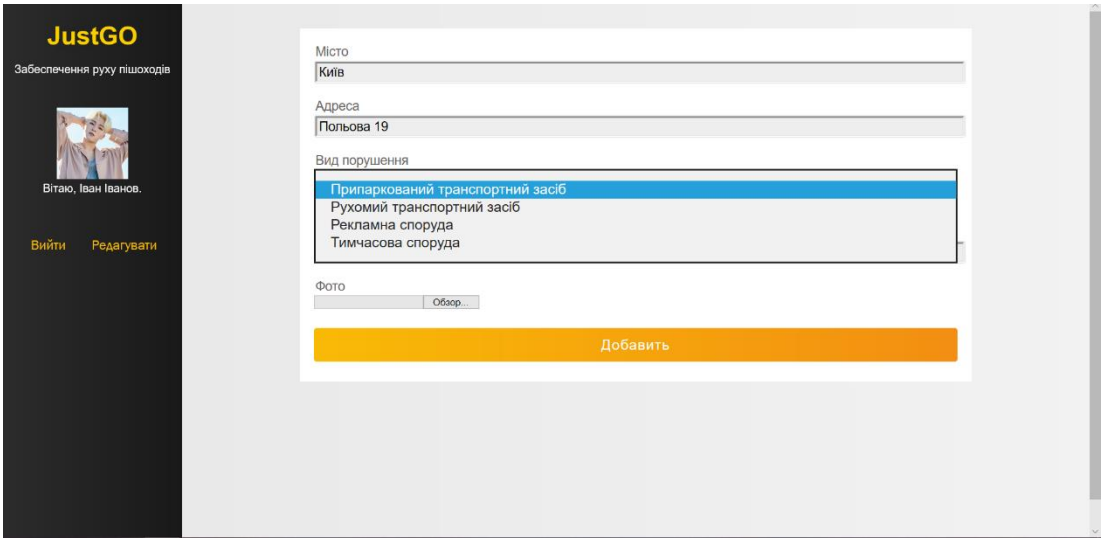


Рис. 3.19. – Види порушення.

Якщо користувач має права адміністратора, веб-додаток відображає йому зовсім іншу сторінку. В цій сторінці адміністратор може переглядати подані заявки (рис.3.20).

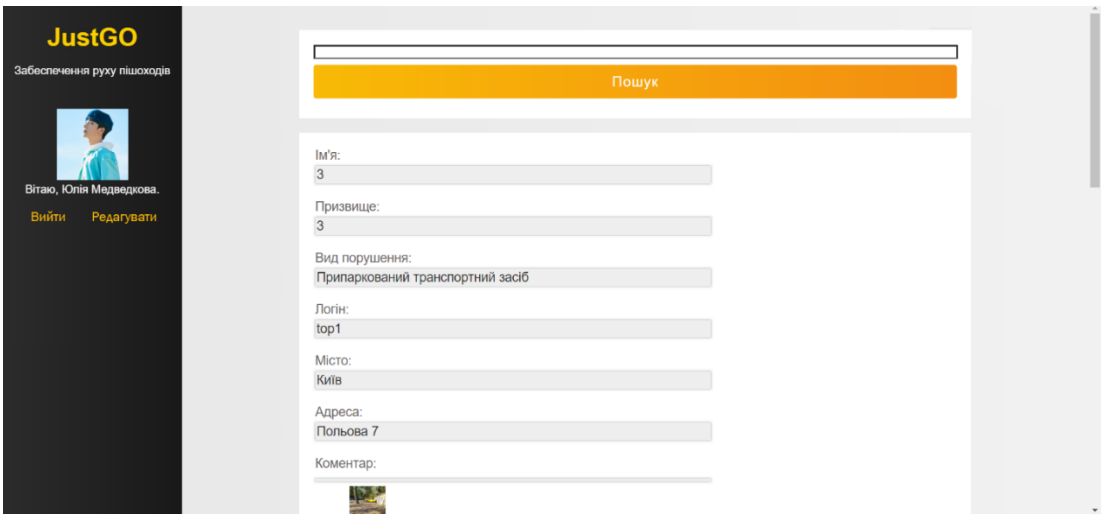


Рис. 3.20. – Сторінка користувача з правами адміністратора.

Щоб адміністратору було легше переглядати подані заявки, було вирішено розробити функцію пошуку. За допомогою пошуку він може шукати заявки по адресі (рис.3.21). Щоб після пошуку відобразити усі заявки, адміністратору потрібно задати пошук з пустим запитом.

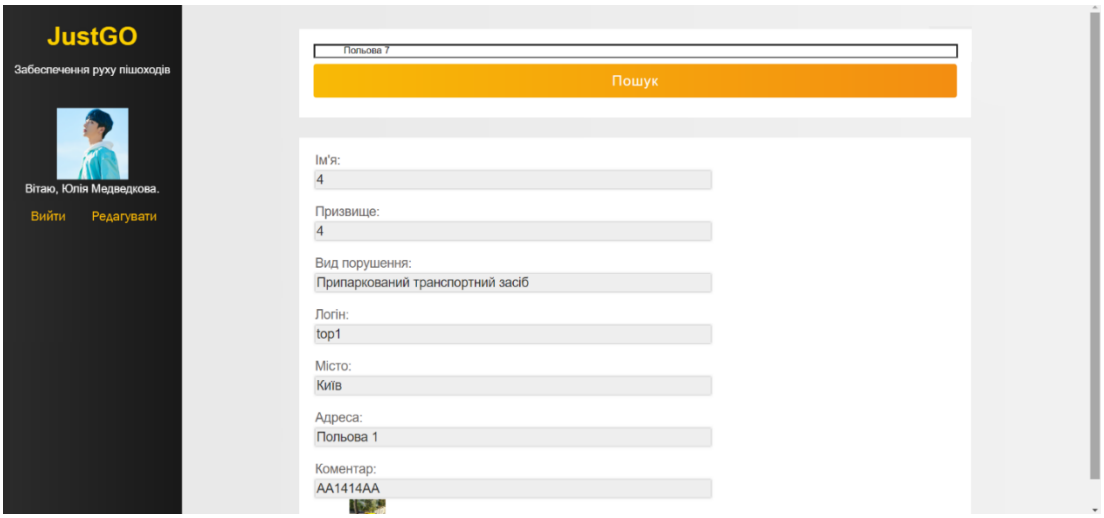


Рис. 3.21. – Пошук заявок по адресі.

Вже було сказано, що сторінка авторизованого користувача відрізняється наявністю блоку користувача. В цьому блоці відображається

зображення користувача, його ім’я та прізвище. А також два посилання – вихід з системи та редагування. Редагування надає можливість користувачу змінити свої дані. Адже, за час існування додатку користувач може змінити ім’я, прізвище, номер телефону, а також логін і пароль (рис.3.22).

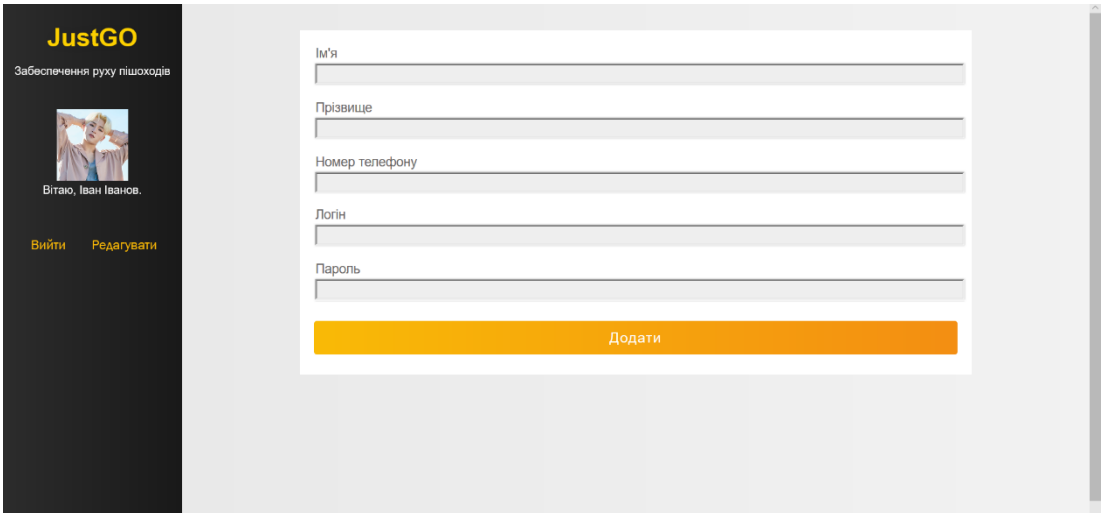


Рис. 3.22. – Редагування сторінки.

Було розглянуто додаток, відображений в комп’ютерному пристрою. Але крім комп’ютерного пристрою додаток налаштовується під розмір будь-який пристрій. Тож, тепер розглянемо додаток, відображений в браузері смартфону.

Як і завжди нас зустрічає головна сторінка, яка пропонує користувачеві увійти в особистий кабінет або зареєструватися.

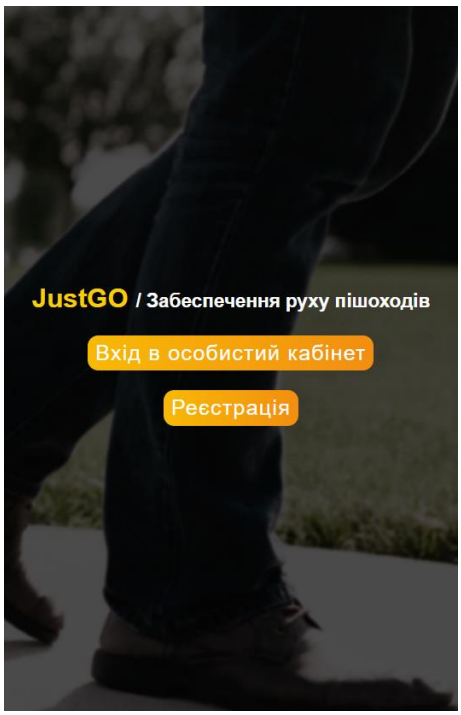


Рис. 3.23. – Головна сторінка веб-додатку в смартфоні.

В реєстрації так само працює обробка введених даних.

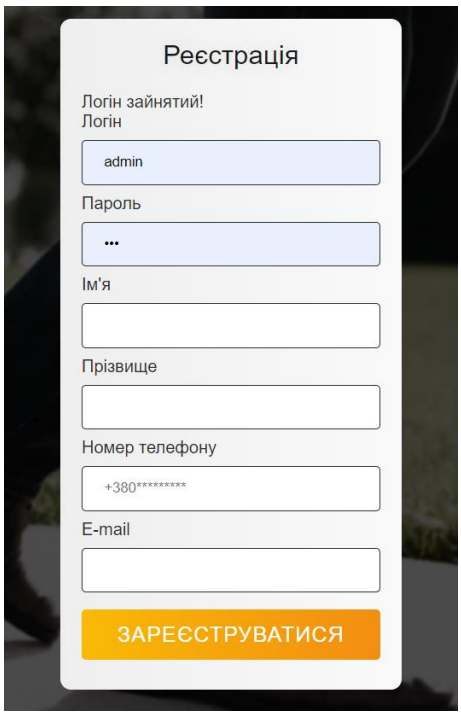


Рис. 3.24. – Помилка в реєстрації веб-додатку, зайнятий логін в смартфоні.

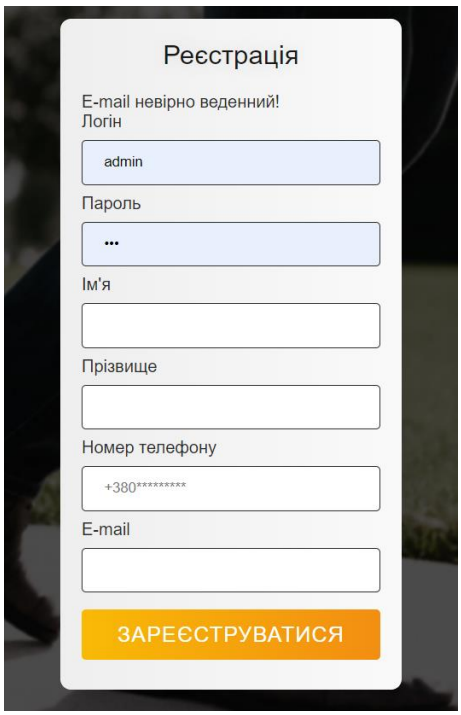


Рис. 3.25. – Помилка в реєстрації веб-додатку, невірний e-mail в смартфоні.

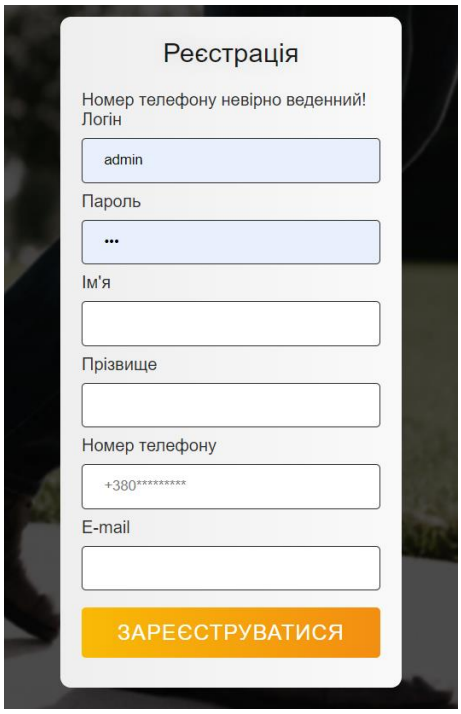


Рис. 3.26. – Помилка в реєстрації веб-додатку, невірний номер телефону в смартфоні.

В авторизації користувач також отримує помилку, якщо додаток не знаходить в базі даних логіна, або якщо пароль не збігається з паролем логіна.

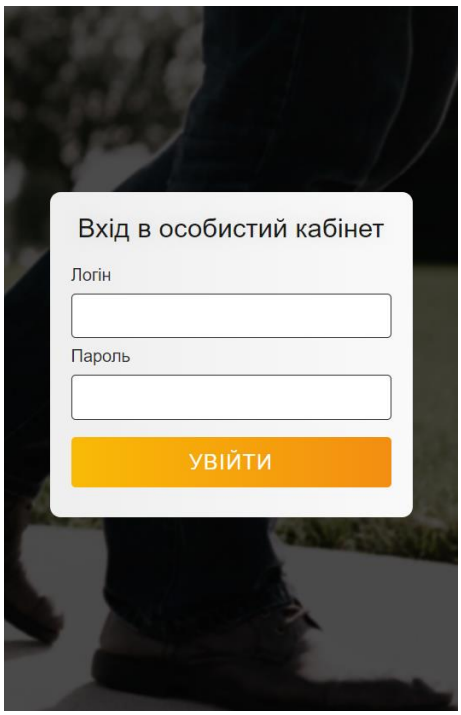


Рис. 3.27. – Вхід в особистий кабінет в смартфоні.

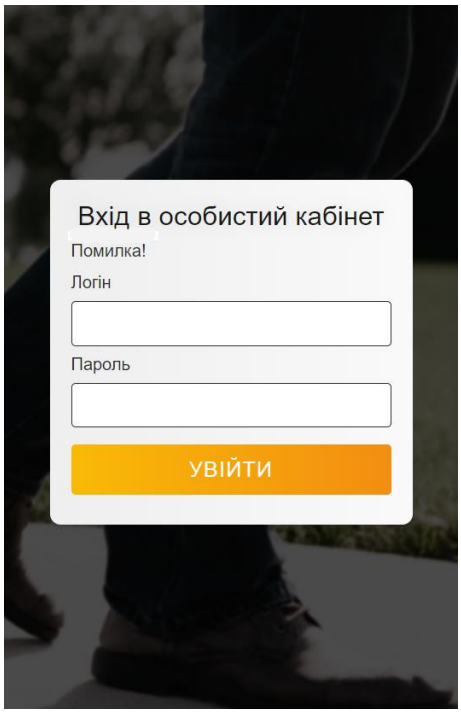


Рис. 3.28. – Помилка у вході в особистий кабінет.

На рис.3.29 видно продемонстрована оновлена сторінка користувача.

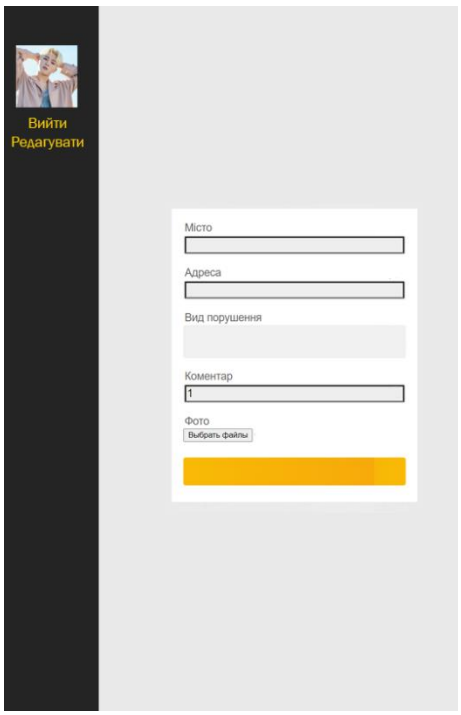


Рис. 3.29. – Сторінка користувача в смартфоні

ВИСНОВОК ДО 3 РОЗДІЛУ

В цьому розділі були розглянуті файли проекту. Це дозволило в цілому відобразити весь реалізований функціонал додатку та показати загальну картину. В даному розділі було розроблено веб-додаток для забезпечення безпеки руху пішоходів. Завдяки SpringMVC вийшло повністю реалізувати MVC шаблон. Тепер можна контролювати HTML-сторінками в повній мірі. В розділі було також розглянуто структуру проекту, а також було відображено весь реалізований функціонал додатку. Також був розроблений зручний інтерфейс, який адаптується під розмір пристрою користувача.

ВИСНОВОКИ

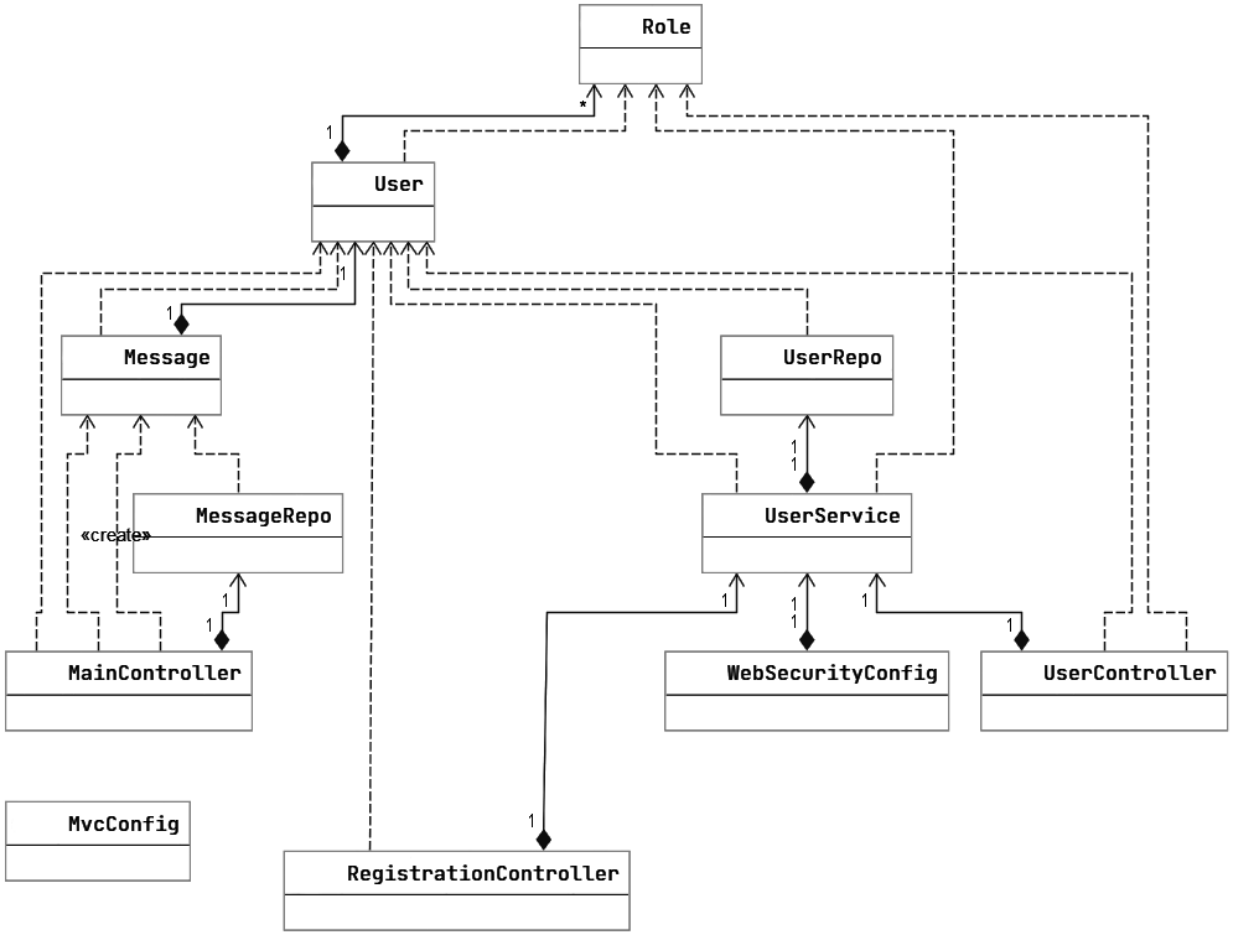
В результаті дипломного проекту було розроблено веб-додаток для забезпечення руху пішоходів. Завдяки фреймворку Spring було реалізовано MVC шаблон, який надав змогу під’єднатися до бази даних, завдяки чому користувачі можуть реєструватися в додатку та заходити в нього, щоб подати заявку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

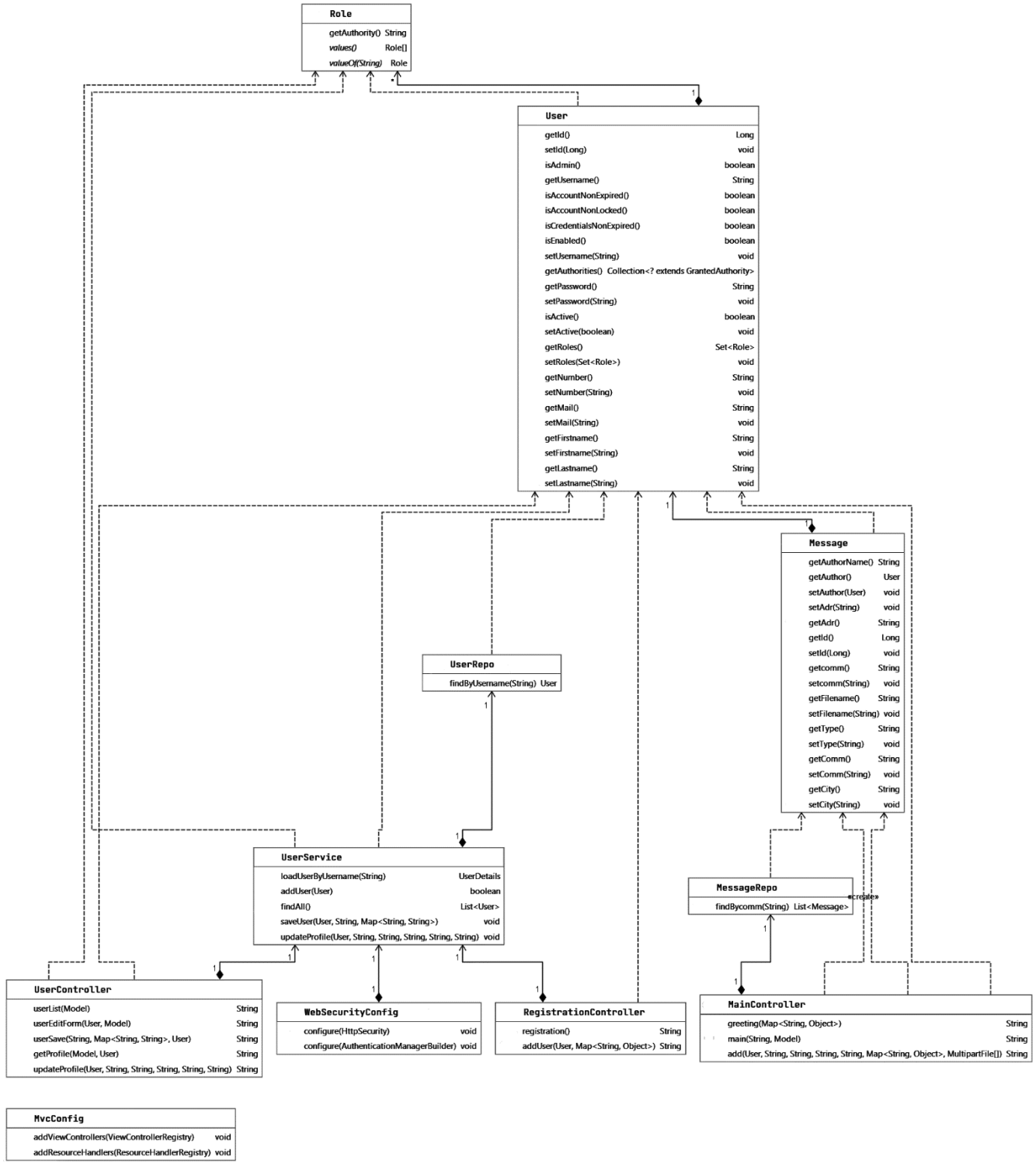
1. Статический сайт [Електронний ресурс] – 2019 – Режим доступу до ресурсу: https://puzzleweb.ru/php/00_teacher.php.
2. Динамический сайт [Електронний ресурс] – 2019 – Режим доступу до ресурсу: https://puzzleweb.ru/php/00_teacher2.php.
3. The Web Application Hacker’s Handbook: Finding and Exploiting Security Flaws, Second Edition / Dafydd Stuttard and Marcus Pinto – Canada, 2006 - 878p.
4. Современные технологии разработки веб-приложений / Д.В. Вагин, Р.В. Петров – Новосибирск, 2019 – 50 с.
5. Web Application Architecture Principles, protocols and practices / John Wiley & Sons Ltd – London, 2003 – 357p.
6. Web Application Architecture: Definition, Models, Types, and More [Електронний ресурс] – 2020 – Режим доступу до ресурсу: https://hackr.io/blog/web-application-architecture-definition-models-types-and-more.
7. Web Application Architecture [Електронний ресурс] – 2018 – Режим доступу до ресурсу: https://svitla.com/blog/web-application-architecture.
8. Державно будівельні норми / Міністерство регіонального розвитку будівництва та житлово-комунального носподарства України - Київ, 2018 - 55с.
9. Правила дорожного движения 2020 [Електронний ресурс] – 2020 – Режим доступу до ресурсу: https://vodiy.ua/ru/pdr/.
10. Web Application Architecture: Definition, Models, Types, and More [Електронний ресурс] – 2020 – Режим доступу до ресурсу: https://hackr.io/blog/web-application-architecture-definition-models-types-and-more.
11. Технології розробки веб-додатків [Електронний ресурс] – 2020 – Режим доступу до ресурсу:http://window.edu.ru/resource/031/76031/files/Posobie-InternetAppDevelopment.pdf.

12. Фреймворки в веб-разработке [Електронний ресурс] – 2020 – Режим доступу до ресурсу: https://web-creator.ru/articles/about_frameworks.
13. UML.Проектирование систем реального времени, параллельных и распределенных приложений: Пер. с англ. - М.: ДМК Пресс, 2011. - 704 с.
14. Проектування і розробка web-додатків– 2017 – Режим доступу до ресурсу: https://stud.com.ua/97678/informatika/proektuvannya_dodatkiv.
15. Documentation PostgreSQL[Електронний ресурс] – 2020 – Режим доступу до ресурсу: https://www.postgresql.org/docs/manuals/.

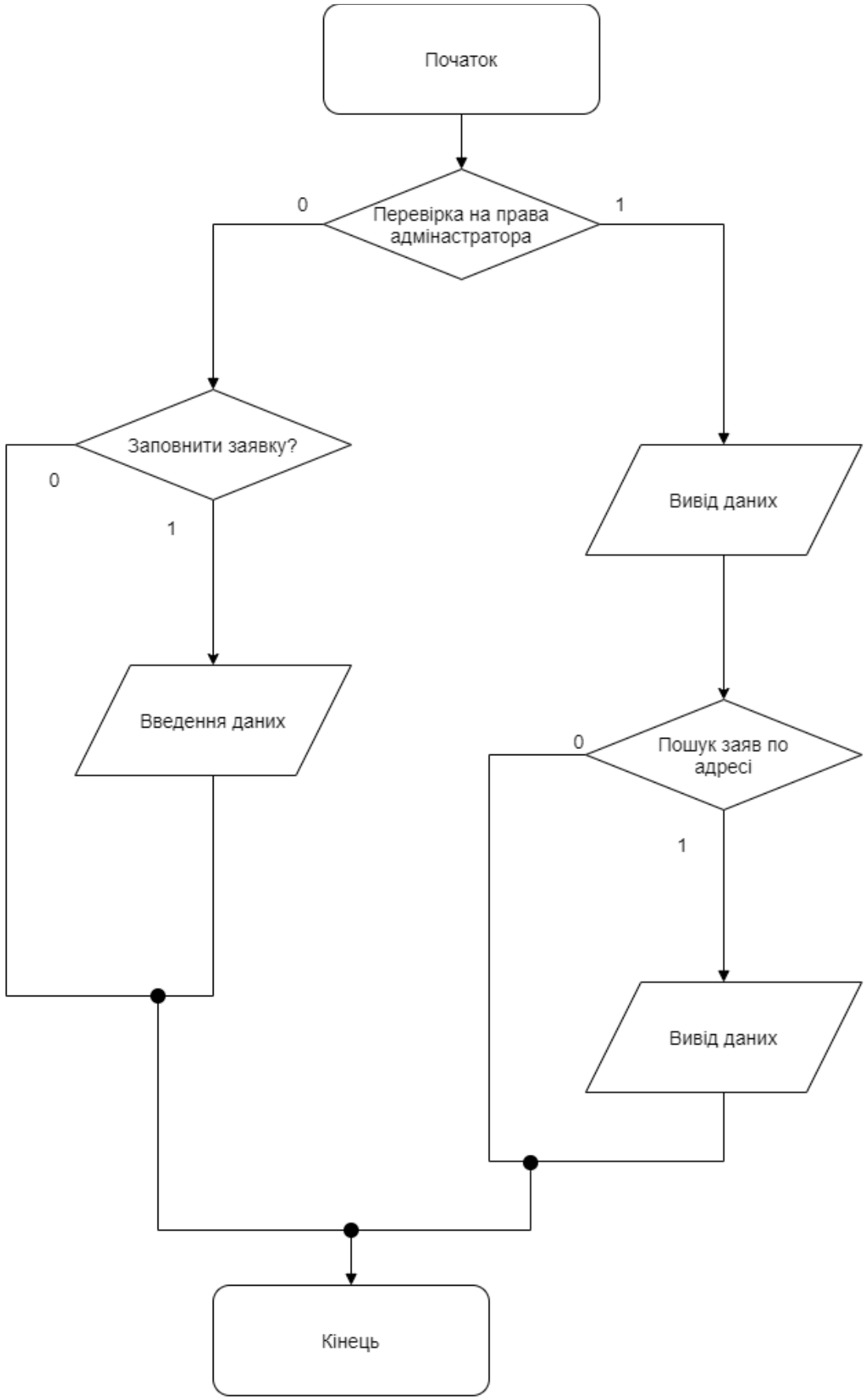
Д О Д А Т К И



					ІАЛЦ.006515.004 Д1		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Медведкова Ю.І.*			**Веб-додаток для безпеки руху пішоходів.** **Структурна схема**	*Літ.*	*Аркуш* / *Аркушів*
Перев.							*1* / **1**
Тех.контр.						***НТУУ "КПІ"ім. Ігоря Сікорського, ФІОТ,ОТ, ІО-62***	
Н.Контр.							
Затвердж.							



					ІАЛЦ.006515.004 Д2	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розроб.		*Медведкова Ю.І.*			**Веб-додаток для безпеки руху пішоходів.** **Функціональна схема**	*Літ.* / *Аркуш* / *Аркушів*: 1 / 1
Перев.						
Тех.контр.						*НТУУ "КПІ"ім. Ігоря Сікорського, ФІОТ,ОТ, ІО-62*
Н.Контр.						
Затвердж.						



					ІАЛЦ.006515.004 Д3		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Медведкова Ю.І.*			**Веб-додаток для безпеки руху пішоходів.** **Принципова схема**	*Літ.*	*Аркуш* / *Аркушів*
Перев.							*1* / **1**
Тех.контр.						***НТУУ "КПІ"ім. Ігоря Сікорського, ФІОТ,ОТ, ІО-62***	
Н.Контр.							
Затвердж.							

V1__Init_DB

```
create sequence hibernate_sequence start 1 increment 1;


create table message (
    id int8 not null,
    filename varchar(255),
    comm varchar(255),
    type varchar(255),
    city varchar(255),
    adr varchar(2048) not null,
    user_id int8,
    primary key (id)
);


create table user_role (
    user_id int8 not null,
    roles varchar(255)
);


create table usr (
    id int8 not null,
    active boolean not null,
    mail varchar(255),
    number varchar(255),
    first_name varchar(255),
    last_name varchar(255),
    password varchar(255) not null,
```

```
    username varchar(255) not null,
    primary key (id)
);

alter table if exists message
    add constraint message_user_fk
    foreign key (user_id) references usr;

alter table if exists user_role
    add constraint user_role_user_fk
    foreign key (user_id) references usr;
```

Application.java

```
package com.example.justgo;


import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

MvcConfig.java

```
package com.example.justgo.config;


import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;


@Configuration
public class MvcConfig implements WebMvcConfigurer {
    @Value("${upload.path}")
    private String uploadPath;
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/login").setViewName("login");
    }


    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/img/**")
                .addResourceLocations("file://" + uploadPath + "/");
        registry.addResourceHandler("/assets/**")
                .addResourceLocations("classpath:/assets/");
    }
}
```

WebSecurityConfig.java

```
package com.example.justgo.config;


import com.example.justgo.service.UserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Configuration;

import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuil
der;

import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurit
y;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;


@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {


    @Autowired

    private UserService userService;


    @Override

    protected void configure(HttpSecurity http) throws Exception {
```

```
        http
                .authorizeRequests()
                .antMatchers("/", "/registration", "/assets/**").permitAll()
                .anyRequest().authenticated()
                .and()
                    .formLogin()
                    .loginPage("/login")
                    .permitAll()
                .and()
                    .logout()
                    .logoutSuccessUrl("/")
                    .permitAll();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userService)
                .passwordEncoder(NoOpPasswordEncoder.getInstance());
    }
}
```

MainController.java

```
package com.example.justgo.controller;


import com.example.justgo.domain.Message;
import com.example.justgo.domain.User;
import com.example.justgo.repos.MessageRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;


import java.io.File;
import java.io.IOException;
import java.util.Map;
import java.util.UUID;


@Controller
public class MainController {
    @Autowired
    private MessageRepo messageRepo;
```

```
    @Value("${upload.path}")
    private String uploadPath;


    @GetMapping("/")
    public String greeting(Map<String, Object> model) {
        return "greeting";
    }


    @GetMapping("/main")
    public String main(@RequestParam(required = false, defaultValue = "") String filter, Model
model) {
        Iterable<Message> messages = messageRepo.findAll();


        if (filter != null && !filter.isEmpty()) {
            messages = messageRepo.findBycomm(filter);
        } else {
            messages = messageRepo.findAll();
        }


        model.addAttribute("messages", messages);
        model.addAttribute("filter", filter);
        return "main";
    }


    @PostMapping("/main")
    public String add(
            @AuthenticationPrincipal User user,
```

```
        @RequestParam String city,
        @RequestParam String type,
        @RequestParam String adr,
        @RequestParam String comm, Map<String, Object> model,
        @RequestParam("file") MultipartFile[] file
    ) throws IOException {
        Message message = new Message(city, adr, type, comm, user);
        System.out.println("----"+user.getFirstname());
        for(MultipartFile f : file) {
            if (f != null && !f.getOriginalFilename().isEmpty()) {
                File uploadDir = new File(uploadPath);

                if (!uploadDir.exists()) {
                    uploadDir.mkdir();
                }

                String uuidFile = UUID.randomUUID().toString();
                String resultFilename = uuidFile + "." + f.getOriginalFilename();

                f.transferTo(new File(uploadPath + "/" + resultFilename));

                message.setFilename(resultFilename);
            }

            messageRepo.save(message);
```

```
            Iterable<Message> messages = messageRepo.findAll();
            model.put("messages", messages);
        }


        return "main";
    }
}
```

RegistrationController.java

```
package com.example.justgo.controller;


import com.example.justgo.domain.User;
import com.example.justgo.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;


import java.util.Map;


@Controller
public class RegistrationController {
    @Autowired
    private UserService userService;


    @GetMapping("/registration")
    public String registration() {
        return "registration";
    }


    @PostMapping("/registration")
    public String addUser(User user, Map<String, Object> model) {


        if (!userService.addUser(user)) {
```

```
            model.put("message", "Логін зайнятий!");
            return "registration";
        }
        return "redirect:/login";
    }
}
```

UserController.java

```
package com.example.justgo.controller;


import com.example.justgo.domain.Role;
import com.example.justgo.domain.User;
import com.example.justgo.repos.UserRepo;
import com.example.justgo.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;


import java.util.Arrays;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;


@Controller
@RequestMapping("/user")


public class UserController {
    @Autowired
    private UserService userService;
```

```
@PreAuthorize("hasAuthority('ADMIN')")
@GetMapping
public String userList(Model model) {
    model.addAttribute("users", userService.findAll());

    return "userList";
}

@PreAuthorize("hasAuthority('ADMIN')")
@GetMapping("{user}")
public String userEditForm(@PathVariable User user, Model model) {
    model.addAttribute("user", user);
    model.addAttribute("roles", Role.values());

    return "userEdit";
}

@PreAuthorize("hasAuthority('ADMIN')")
@PostMapping
public String userSave(
        @RequestParam String username,
        @RequestParam Map<String, String> form,
        @RequestParam("userId") User user
) {
    userService.saveUser(user, username, form);
```

```
        return "redirect:/user";
    }

    @GetMapping("profile")
    public String getProfile(Model model, @AuthenticationPrincipal User user){
        model.addAttribute("username", user.getUsername());
        model.addAttribute("first_name", user.getFirstname());
        model.addAttribute("mail", user.getMail());
        model.addAttribute("number", user.getNumber());
        return "profile";
    }

    @PostMapping("profile")
    public String updateProfile(
            @AuthenticationPrincipal User user,
            @RequestParam String first_name,
            @RequestParam String last_name,
            @RequestParam String password,
            @RequestParam String number,
            @RequestParam String mail
        ){
        userService.updateProfile(user, first_name, last_name, number, password, mail);
        return "redirect:/user/profile";
    }
}
```

Message.java

```
package com.example.justgo.domain;


import org.springframework.jmx.export.annotation.ManagedAttribute;


import javax.persistence.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;


@Entity
public class Message {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;


    private String type;
    private String adr;
    private String comm;
    private String city;


    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_id")
    private User author;
    private String filename;
```

```
public Message() {
}

public Message(String city, String adr, String type, String comm, User user) {
    this.city = city;
    this.adr = adr;
    this.type = type;
    this.author = user;
    this.comm = comm;
}

public String getAuthorName() {
    return author != null ? author.getUsername() : "<none>";
}

public User getAuthor() {
    return author;
}

public void setAuthor(User author) {
    this.author = author;
}

public void setAdr(String adr) {
    this.adr = adr;
}
```

```
public String getAdr() { return adr; }

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getcomm() {
    return comm;
}

public void setcomm(String comm) { this.comm = comm; }

public String getFilename() {
    return filename;
}

public void setFilename(String filename) {
    this.filename = filename;
}

public String getType() {
```

```
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getComm() {
        return comm;
    }

    public void setComm(String comm) {
        this.comm = comm;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }
}
```

Role.java

```
package com.example.justgo.domain;


import org.springframework.security.core.GrantedAuthority;


public enum Role implements GrantedAuthority {
    USER, ADMIN;


    @Override
    public String getAuthority() {
        return name();
    }
}
```

User.java

```
package com.example.justgo.domain;


import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;


import javax.persistence.*;
import java.util.Collection;
import java.util.Set;


@Entity
@Table(name = "usr")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String username;
    private String password;
    private boolean active;


    private String number, mail, first_name, last_name;


    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;
```

```
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public boolean isAdmin(){
    return roles.contains(Role.ADMIN);
}

public String getUsername() {
    return username;
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}
```

```
@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return isActive();
}

public void setUsername(String username) {
    this.username = username;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return getRoles();
}

public String getPassword() { return password; }

public void setPassword(String password) {
    this.password = password;
}
```

```
public boolean isActive() {
    return active;
}

public void setActive(boolean active) {
    this.active = active;
}

public Set<Role> getRoles() {
    return roles;
}

public void setRoles(Set<Role> roles) {
    this.roles = roles;
}

public String getNumber() {
    return number;
}

public void setNumber(String number) { this.number = number; }

public String getMail() {
    return mail;
}
```

```
    public void setMail(String mail) {
        this.mail = mail;
    }

    public String getFirstname() {
        return first_name;
    }

    public void setFirstname(String first_name) {
        this.first_name = first_name;
    }

    public String getLastname() {
        return last_name;
    }

    public void setLastname(String lastname) {
        this.last_name = lastname;
    }
}
```